

NOMBRE DE LA CLASE:

Programación en hoja cuadriculada

Duración: 45 - 60 minutos : Preparación: 10 minutos

Meta: Ayudar a los estudiantes a entender qué significa “programar”.

RESUMEN:

Programando algunos dibujos, los estudiantes empezarán a entender de qué trata la programación. La clase empezará haciendo que los estudiantes intercambien instrucciones para colorear cuadrados en una hoja cuadriculada, con el fin de reproducir una imagen en la hoja. La clase puede terminar exhibiendo los dibujos que los estudiantes realizaron.

OBJETIVO:

Los estudiantes podrán —

- Entender la dificultad de traducir problemas reales en programas
- Aprender que las ideas pueden parecer muy claras para ellos, pero mal interpretadas por una computadora
- Comprender la necesidad de estructuras formales de programación como repeticiones y funciones.

MATERIALS:

- Kit de dibujos y algoritmos de ejemplo
- Tarjetas de programación
- Hojas cuadriculadas grandes

- Fibras, lapiceras y lápices (dos o tres colores)

PREPARACIÓN:

Imprimir el Kit de dibujos y algoritmos para cada grupo.

Imprimir las Tarjetas de programación para cada grupo.

Proveer a cada grupo varias hojas cuadriculadas.

VOCABULARIO:

Algoritmo—Una serie de instrucciones que permite ejecutar una tarea

Programar/Codificar—Transformar acciones a un lenguaje simbólico

Depurar—Encontrar y arreglar problemas en el código de los programas

Función—Una pieza de código que puedes ser ejecutada tantas veces como queramos

Parámetros—Datos adicionales que podemos pasar a las funciones para adaptarlas a nuestras necesidades

REPASO:

La intención de este repaso es recordar lo visto en la clase anterior. Si cubres las actividades en distinto orden, por favor realiza tu propio repaso aquí.

Preguntas para la participación en clase:

- ¿Puedes nombrar alguno de los pasos del pensamiento computacional?
- ¿Puedes recordar alguno de los patrones que encontramos en los monstruos de la última clase?

Debate:

- ¿Qué otra cosa podemos describir con las mismas partes abstraídas en la clase de los monstruos? ¿Podemos describir una vaca? ¿Un pájaro? ¿Debemos cambiar algo para poder describir una tetera?



Programando algunos dibujos, los estudiantes empezarán a entender de qué trata la programación.

DESARROLLO:













Empezar preguntando a la clase si alguno escuchó hablar de robótica. **¿Qué es un robot?**
¿Un robot realmente puede “entender” lo que la gente dice? La respuesta a esta última pregunta es la siguiente:

“No de la misma forma en que lo entiende una persona.”

Los robots operan con “instrucciones”, conjuntos específicos de acciones que pueden realizar porque fueron programados para poder hacerlas. Para poder llevar a cabo una tarea, un robot necesita tener una serie de instrucciones (a veces llamadas “algoritmos”) que puedan ejecutar.

Para familiarizarnos más con el concepto de algoritmo, sirve poder compararlo con otras cosas. Para este ejercicio, vamos a presentar un lenguaje de programación hecho de líneas y flechas.

SÍMBOLOS PARA PROGRAMAR

- | | | |
|---|---|---|
|  |  | Mover un cuadrado adelante |
|  |  | Mover un cuadrado atrás |
|  |  | Mover un cuadrado arriba |
|  |  | Mover un cuadrado abajo |
|  |  | Cambiar al siguiente color |
|  |  | Pintar cuadrado con el color seleccionado |
-

En este ejemplo, los símbolos de la izquierda son el “programa” y las palabras de la derecha son los “algoritmos”. Esto significa que podemos escribir el siguiente algoritmo:

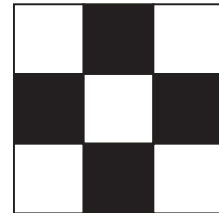
“Mover un cuadrado adelante, mover un cuadrado adelante, pintar el cuadrado con el color seleccionado”

y esto se corresponde con el siguiente programa:



Ahora es tiempo de practicar un poco. Inicia la clase en el mundo de la programación dibujando o proyectando el siguiente ejemplo en el pizarrón.

Selecciona un dibujo simple como el siguiente para usar de ejemplo:



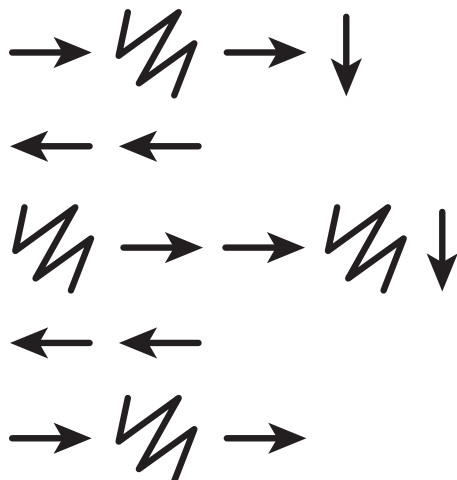
Esta es una buena manera de introducir todos los símbolos del lenguaje de programación. Para empezar, puedes mostrarles una forma menos confusa de codificar una imagen, que es retornar a la izquierda de la imagen siempre que terminas con la línea siguiente.

Llenar la cuadrícula para la clase y luego pregúntales qué acabas de hacer. Primero puedes poner las palabras en forma de algoritmo, para que luego ellos deduzcan el programa.

Ejemplo de algoritmo:

**“adelante, pintar cuadrado, adelante, línea siguiente,
atrás, atrás,
pintar cuadrado, adelante, adelante, pintar cuadrado, línea siguiente,
atrás, atrás,
adelante, pintar cuadrado, adelante”**

Algunos en la clase pueden notar que hacemos algunos pasos innecesarios, pero intenta frenarlos hasta que llegue la parte de programar. Pasar a programar la imagen anteriormente descrita:

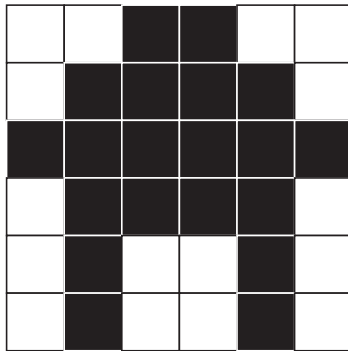
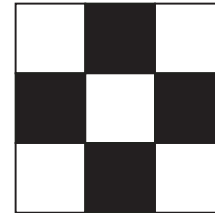


La clase puede realizar algunas preguntas a este punto. Si la clase entiende el objetivo de este ejercicio, entonces puede ser tiempo de entretenerlos.

Es verdad que hacemos algunos pasos innecesarios en el algoritmo anterior, pero puede resultar un código confuso si quitamos esos pasos. A veces, hasta que un programador tenga suficiente experiencia, es recomendable que realice los programas de una forma en la que se entienda que funcionan correctamente, y luego se pueden borrar los pasos innecesarios.

Trabajar otra vez sobre el ejemplo dado, primero quitando los pasos innecesarios del programa original, y luego codificando el dibujo desde cero, directamente sin utilizar símbolos de más.

Si la clase se confunde, intenta con otro gráfico pero elimina la parte de simplificación. Sólo sigue los pasos de izquierda a derecha una y otra vez.

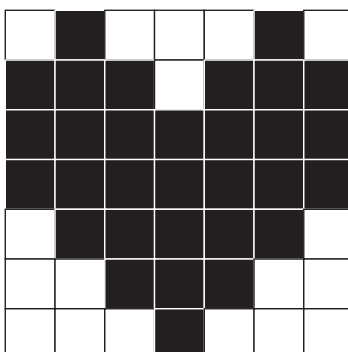


Si la clase puede descifrar el algoritmo, entonces define los símbolos a usar en cada paso, porque están listos para continuar con algo más complejo. Dependiendo de tu clase y la edad que tienen, puedes probar ejercicios más difíciles y permitirles trabajar en grupos.

Dar a cada grupo un grupo de imágenes y sugerir que elijan una imagen monocromática. Cuando definan su primer algoritmo, entonces pedir que lo pasen a símbolos. Una vez que hicieron dos o tres, intercambiar algoritmos entre grupos y pedirles que sigan el programa para dibujar lo que otros programaron.

Si pudieron resolverlos exitosamente, entonces podemos introducir funciones y parámetros. De esta manera se pueden completar dibujos más grandes y complicados.

Luego de dibujar uno de estas imágenes, es obvio que una imagen de este tamaño es bastante tediosa. Miremos sólo dos líneas de este dibujo.



Algoritmo:

“Adelante, pintar cuadrado, adelante, adelante, adelante, pintar cuadrado, adelante, siguiente línea.

Atrás, atrás, atrás, atrás, atrás, atrás.

Pintar cuadrado, adelante, pintar cuadrado, adelante, pintar cuadrado, adelante, adelante, pintar cuadrado, adelante, pintar cuadrado, siguiente línea.

Atrás, atrás, atrás, atrás, atrás, atrás.”

Desafía a la clase a observar qué cosas se repiten a menudo. Tomar algunas de las sugerencias, pero ciertamente una de las cosas más útiles a combinar es “Atrás, atrás, atrás, atrás, atrás, atrás.” Preguntar a los chicos por sugerencias de cómo volver esto un símbolo más del lenguaje.

La clase puede llegar a crear algo como lo siguiente:

← 6

De hecho, pueden llegar a utilizar algo similar para la última tanda de imágenes. ¿Existen otras combinaciones que podemos crear? ¿Saltar tres cuadrados de una fila? ¿Colorear tres cuadrados consecutivos de una fila? ¿Colorear siete cuadrados de una fila?

Puedes terminar teniendo muchos símbolos que contienen varios números. Esperamos que la clase entienda lo útiles que son.

¡Ahora comienza la magia! ¡Revelas a la clase que descubrieron cómo crear funciones! Crearon una representación simple del agrupamiento de varias acciones. Eso es exactamente el objetivo de las funciones. ¿Y el número que colocamos al lado del símbolo? Eso también tiene un nombre. Ese número se llama “parámetro”. En el ejemplo de arriba, el parámetro le dice a la función cuántas veces moverse hacia atrás.

Mira las siguientes “funciones”. ¿Qué se supone que hacen?

1. (→ 6)

2. (→ ↗ 6)

3. (↗ → ↓ 6)

Respuestas:

1. Se mueve seis espacios hacia delante.
2. Pinta 6 cuadrados en una fila.
3. Pinta una diagonal de 6 cuadrados.

¿Hay otras combinaciones que pueden resultar útiles?

Armados de esta nueva herramienta, el desafío es elegir una de las imágenes más difíciles que puedan manejar. ¿Ayudan estos nuevos símbolos a completar el proceso más rápidamente?

Cuando el grupo complete la tarea, pídeles que intercambien sus programas (incluyendo las nuevas funciones y parámetros) con otro grupo y que traten de dibujar lo que codificaron.

AJUSTES:

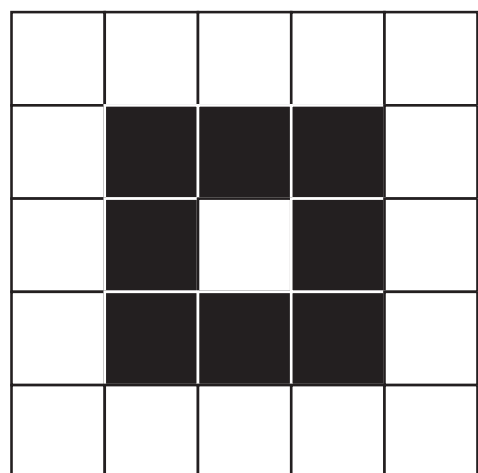
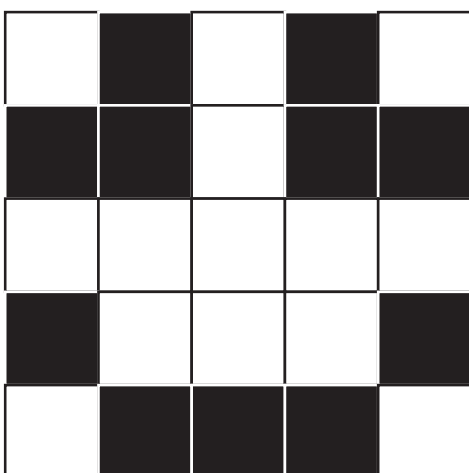
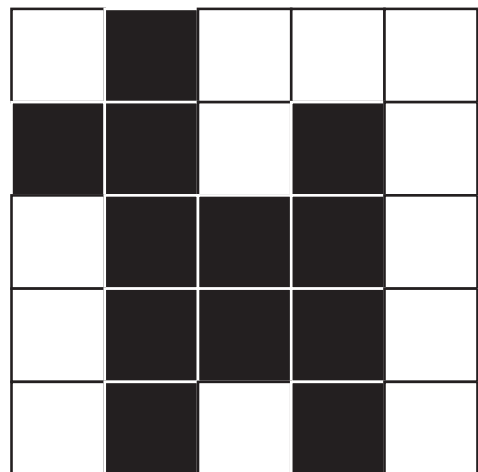
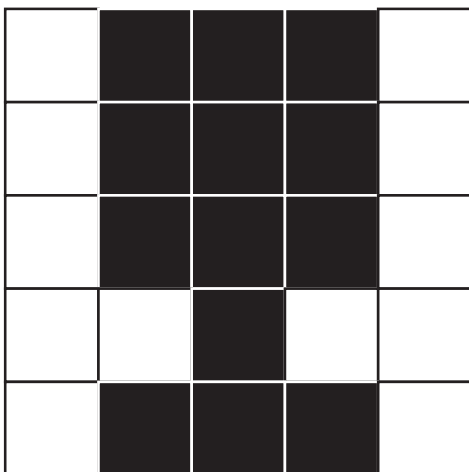
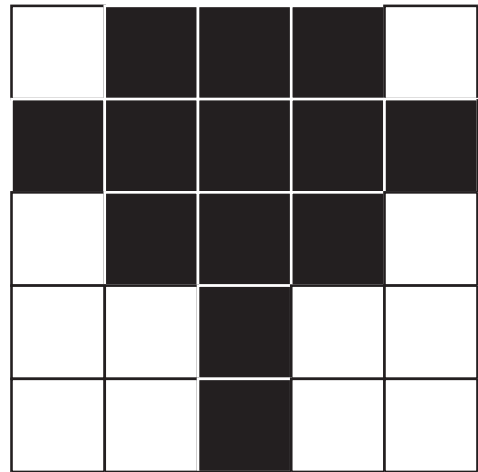
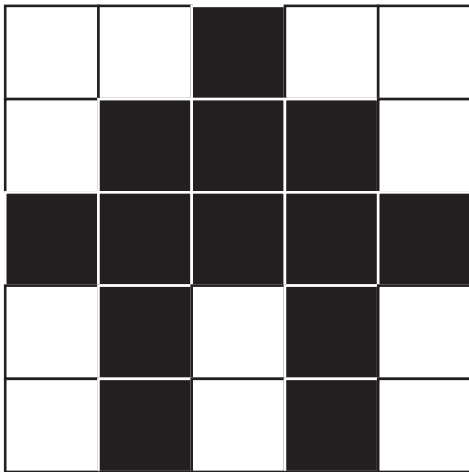
De 5 a 6 años: Toda la clase puede trabajar en la misma imagen y algoritmo. Usar una hoja cuadriculada más grande. Usar un único color para las imágenes.

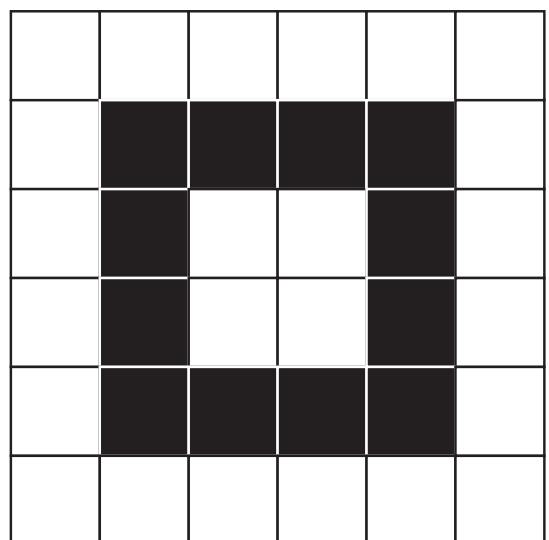
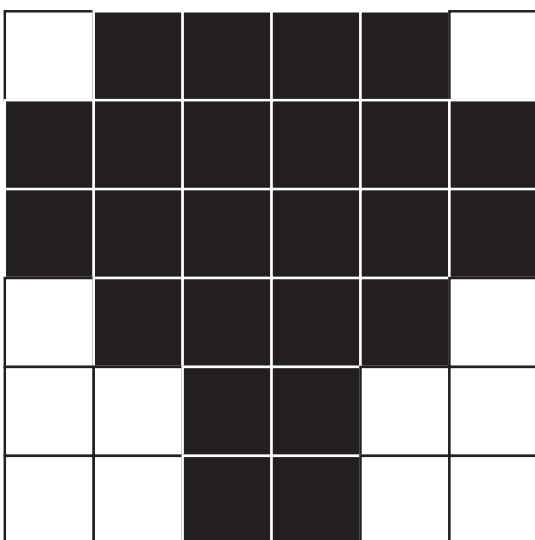
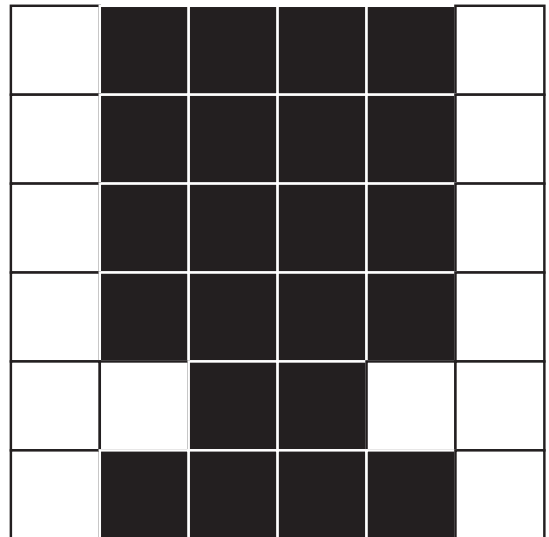
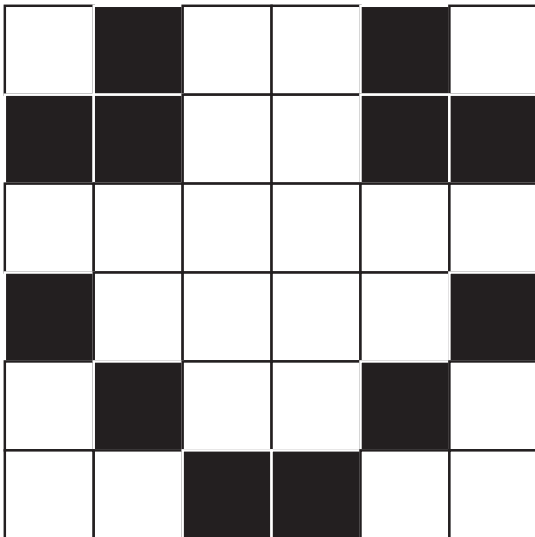
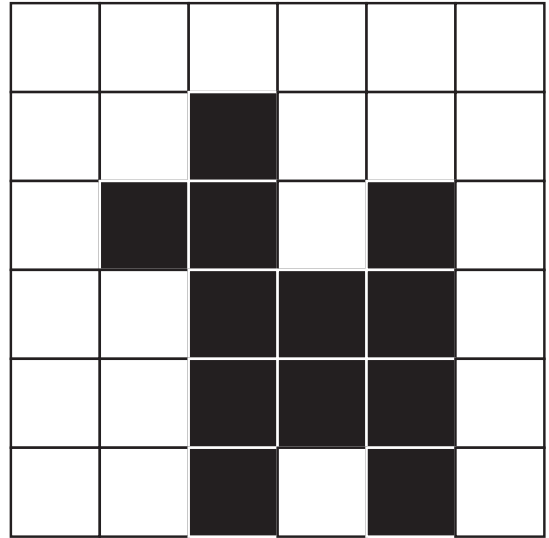
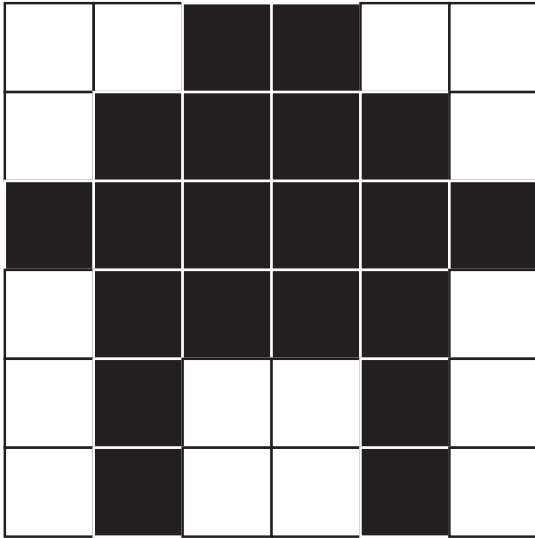
De 7 a 9 años: Trabajar en grupos pequeños (2-4). Usar una hoja más pequeña para agilizar el tiempo que tardan dibujando.

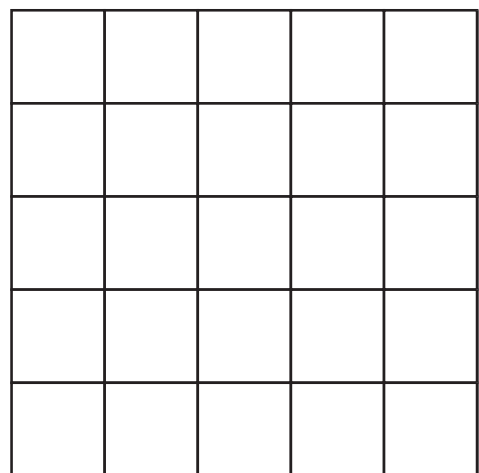
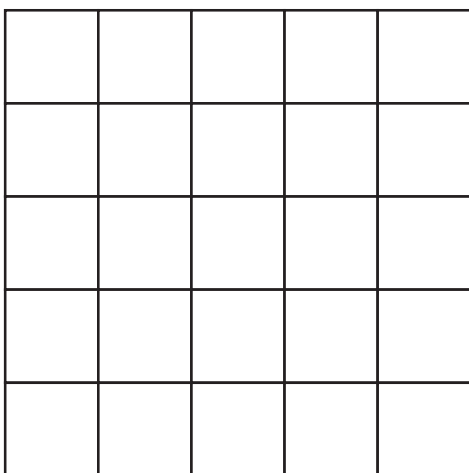
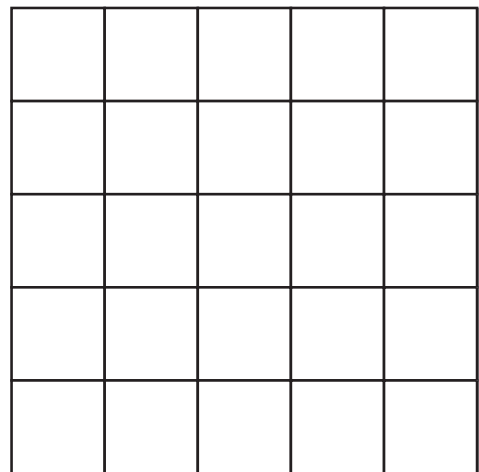
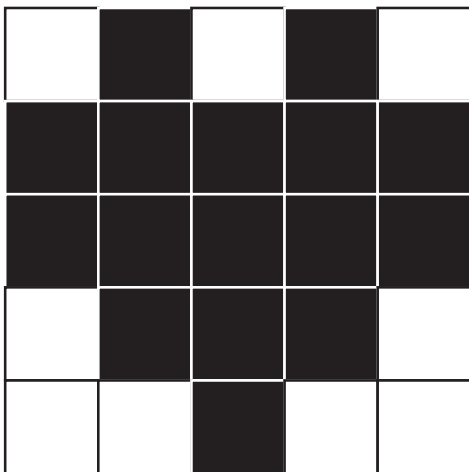
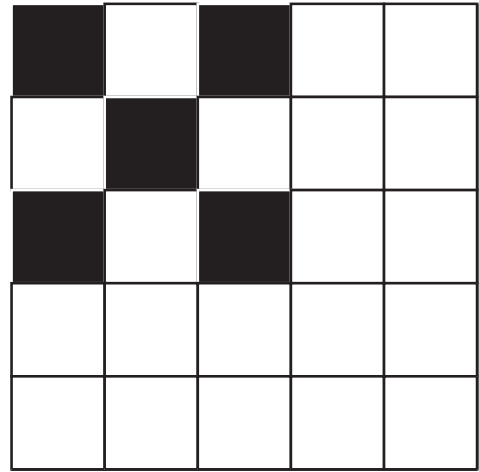
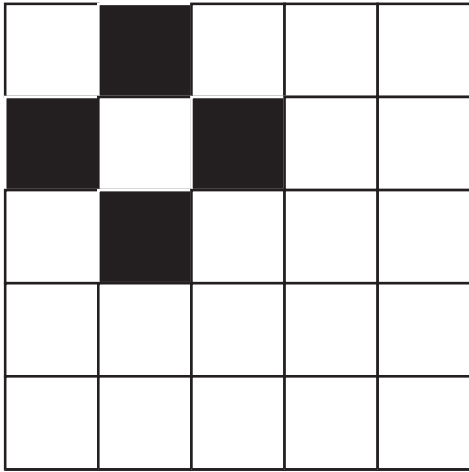
De 10 a 12 años: Trabajar de a pares o individualmente.

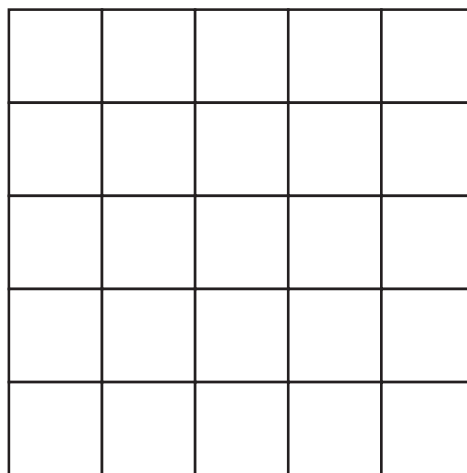
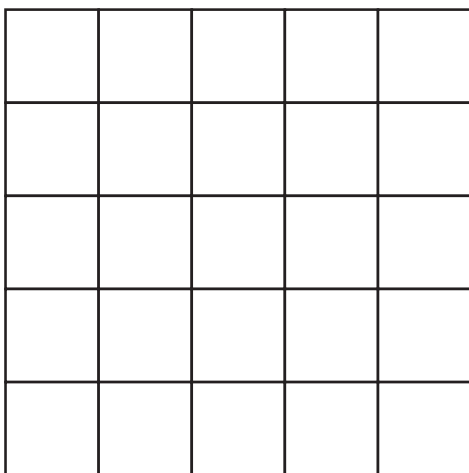
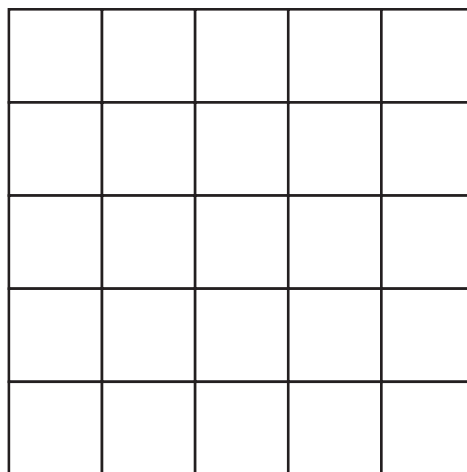
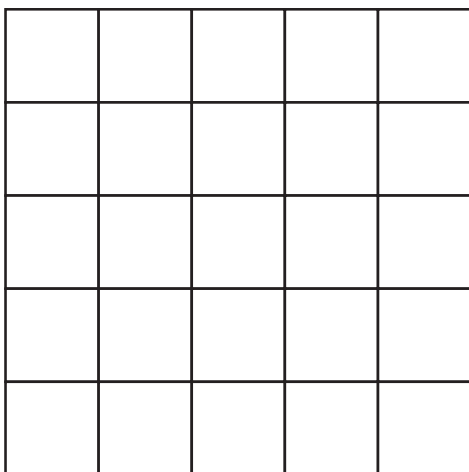
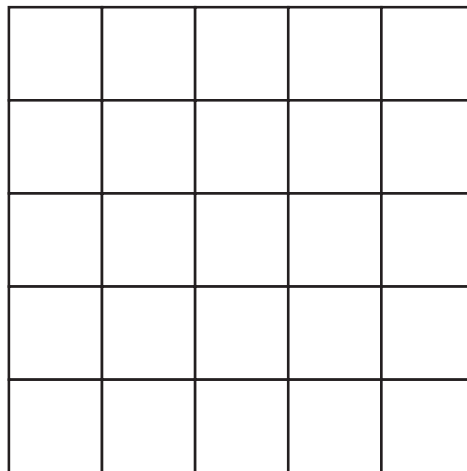
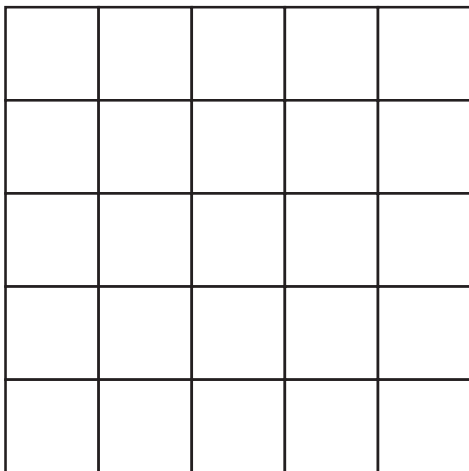
STEPS:

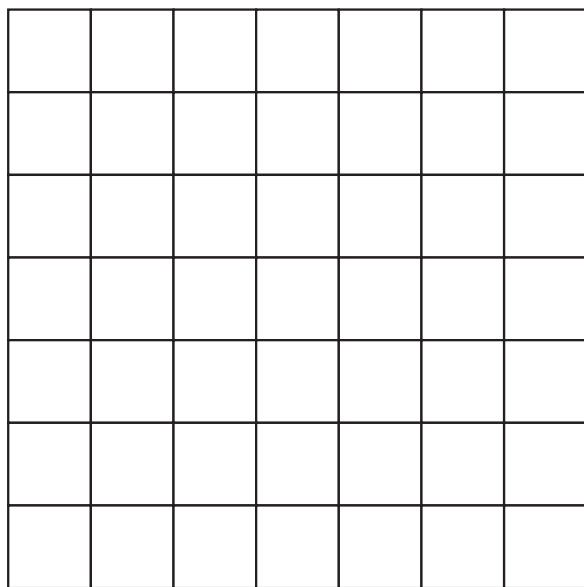
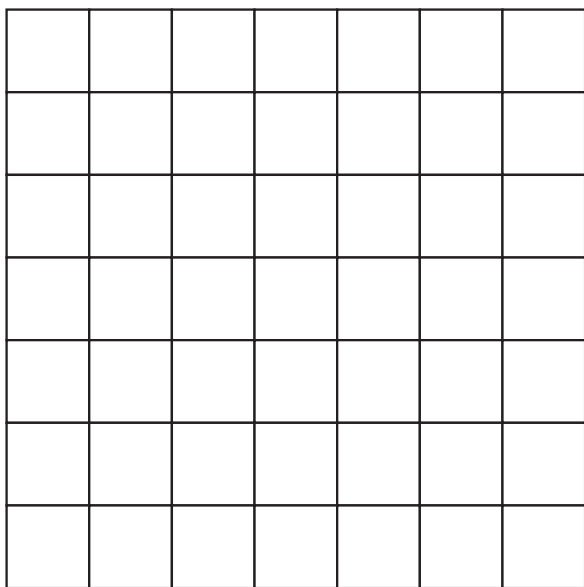
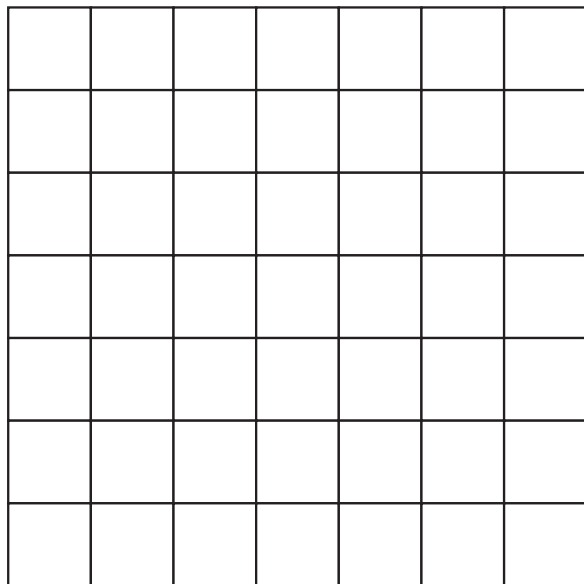
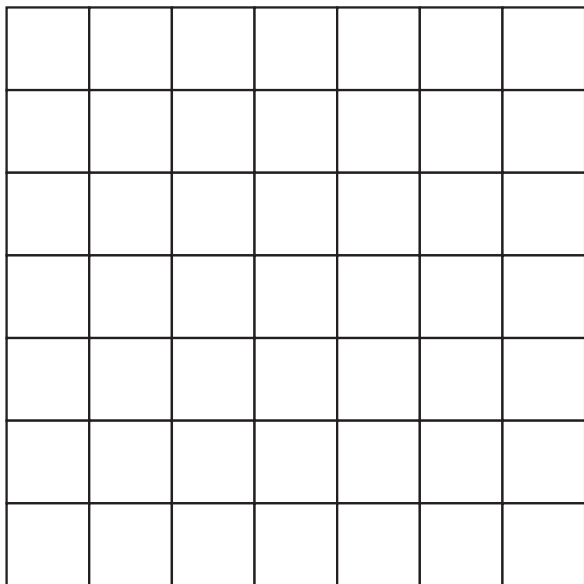
- 1) Elegir una imagen del pack de imágenes.
- 2) Escribir un algoritmo para dibujar esa imagen.
- 3) Convertir el algoritmo en un programa usando símbolos.
- 4) Intercambiar programas con otro equipo y dibujar su imagen.
- 5) Añadir “funciones” para hacer los programas más simples.
- 6) Escribir programas para imágenes más complejas.
- 7) Intercambiar los programas complejos y dibujarlos nuevamente.

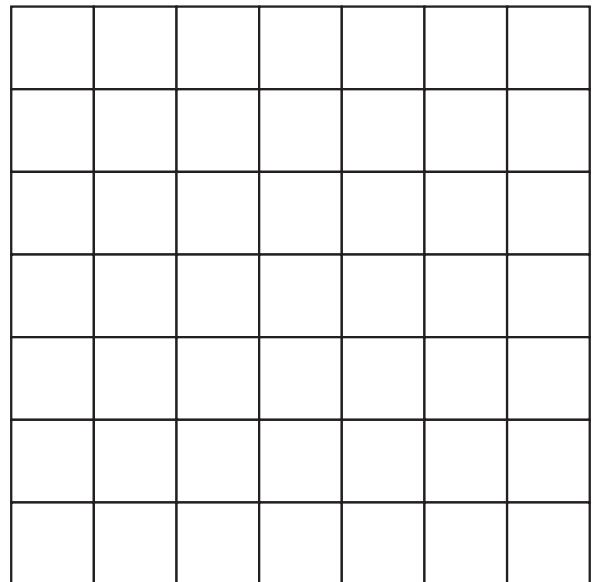
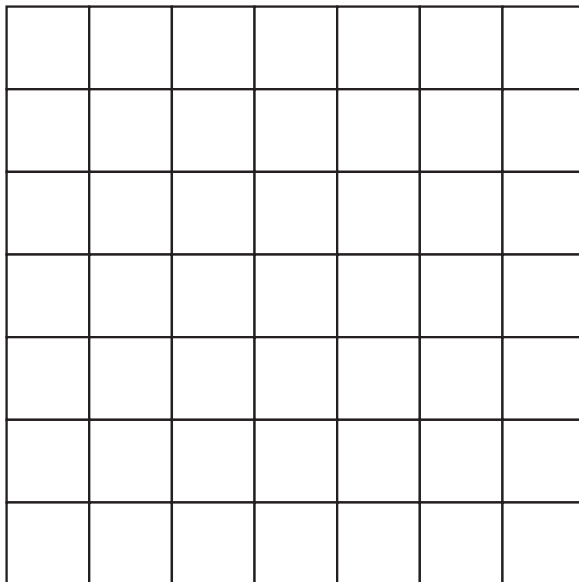
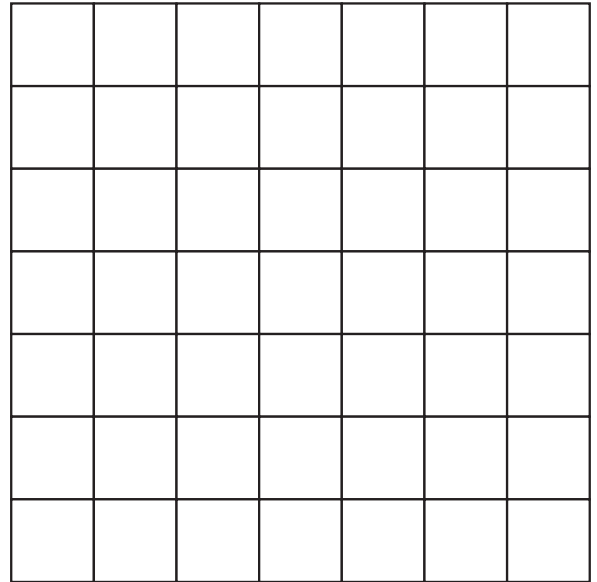
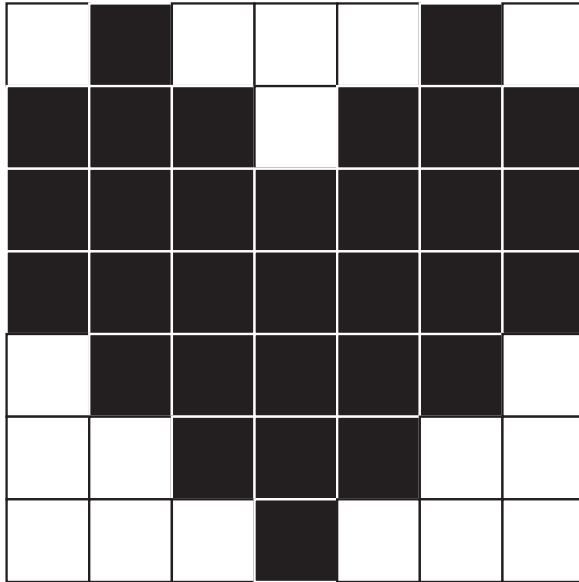


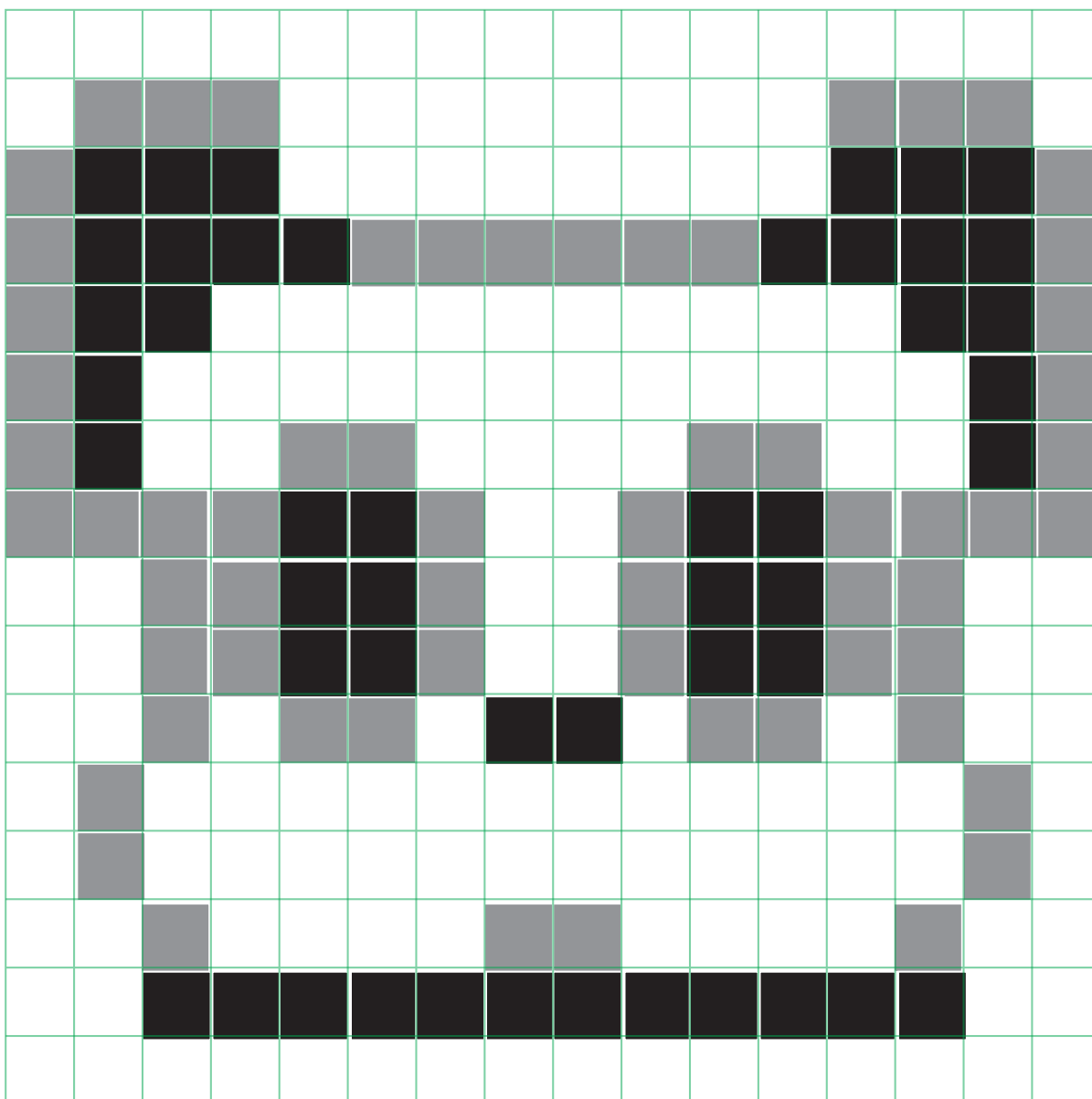


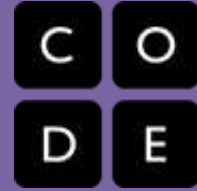












Course 3

OVERVIEW

Students create programs with different kinds of loops, events, functions, and conditions and write algorithms for everyday tasks. Through this they will investigate different problem-solving techniques, discuss societal impacts of computing and the Internet, and learn about Internet transmission methods. By the end of the curriculum, students create interactive stories and games they can share with anyone. Students taking Course 3 will have already taken Course 2.

Lesson Sequence of Course 3

Online lessons are in regular text and unplugged activities are in **bolded** text.

#	Lesson Name	Description
1	Computational Thinking	Students use the steps of computational thinking (decompose, pattern match, abstract, algorithm) to figure out how to play a game that comes with no instructions.

#	Lesson Name	Description
2	Maze	Students write programs (an algorithm for the computer) that get a character through a maze. They'll understand the importance of sequence and basic loops (repeated statements) in the programs they write.
3	Artist	Students write programs to draw different shapes.
4	Functional Suncatchers	Students create an algorithm with functions (pieces of code that you want to use over and over again) to create suncatchers using string and beads.

#	Lesson Name	Description
5	Artist: Functions	Using and modifying prebuilt procedures in the the Artist environment, students gain familiarity with how code is written for functions.
6	Bee: Functions	Using the Bee environment, students use and modify functions to help the bee collect nectar and make honey.
7	Bee: Conditionals	In the Bee environment, students write programs with conditional statements. Students originally learned this concept in Course 2, but this lesson introduces more complex implementations of conditionals.

#	Lesson Name	Description
8	Maze: Conditionals	Using the Maze environment, students write programs using conditionals.
9	Songwriting	Students use the concept of the chorus in a song to learn about functions.
10	Real-Life Algorithms - Dice Race	This lesson calls out ways we use algorithms in our daily lives. Students have to identify and write down the algorithm for a dice race game.
11	Artist: Nested Loops	Students use the Artist environment to write programs that have looped statements inside another loop, which is called a nested loop.

#	Lesson Name	Description
12	Farmer: While Loops	Using while loops, students control a farmer shovel dirt into holes until they're full and remove dirt from piles until it's all gone.
13	Bee: Nested Loops	Students use the Bee environment to write programs using nested loops.
14	Bee: Debugging	Using the same environment as the prior online activity, students are presented with a pre-written program that fails to complete the puzzle. Students will have to "debug" or fix the pre-written program.

#	Lesson Name	Description
15	Bounce	Using the concept of “Events,” (a concept learned in Course 2) students will create a game of their own with events like “When the ball goes through the goal, you score a point.”
16	Play Lab: Create a Story	Students use the Mini-Studio environment to create their own interactive stories.
17	Play Lab: Create a Game	Students use the Mini-Studio environment to create their own interactive games.
18	Internet	Students send messages representing various Internet transmission methods using pieces of paper.

#	Lesson Name	Description
19	Crowdsourcing	Student use crowdsourcing, a problem-solving technique common in computer science, to complete a task together as a classroom more efficiently than if a student attempted it alone.
20	Digital Citizenship	Students explore the difference between private information and personal information, distinguishing what is safe and unsafe to share online.