

LESSON NAME:

# Graph Paper Programming

Lesson time: 45–60 Minutes : Prep time: 10 Minutes

**Main Goal:** Help students understand how “coding” works.

**OVERVIEW:**

By programming one another to draw pictures, students will begin to understand what programming is really about. The class will begin by students instructing each other to color squares in on graph paper in an effort to reproduce an existing picture. If there's time, the lesson can conclude with images that the students create themselves.

**OBJECTIVE:**

Students will —

- Understand the difficulty of translating real problems into programs
- Learn that ideas may feel clear to them, and still be misinterpreted by a computer
- Realize the need for formal programming structures like loops and functions

**MATERIALS:**

- Sample Drawings/Algorithms Kit
- Programming Instructions Card
- Large grid graph paper
- Markers, pens, or pencils (two or three colors)

**PREPARATION:**

Print out a Drawings/Algorithms Kit for each group.

Print Programming Instructions Card for each group.

Supply each group with several drawing grids.

**VOCABULARY:**

**Algorithm**—A series of instructions on how to accomplish a task

**Coding**—Transforming actions into a symbolic language

**Debugging**—Finding and fixing problems in code

**Function**—A piece of code that can be called over and over

**Parameters**—Extra bits of information that you can pass into a function to customize it

**REVIEW:**

This review segment is intended to get the class thinking back to the last lesson. If you are covering these activities out of order, please substitute your own review subjects here.

**Class Participation Questions:**

- Can you name any of the steps of computational thinking?

- Can you remember any of the patterns we found among the monsters of last class?

**Elbow Partner Discussion:**

- What else could we describe with the same “abstracted” concepts from the monster lesson? Could we describe a cow? A bird? Would anything have to change to describe a teapot?



By programming one another to draw pictures, students will begin to understand what programming is really about.

**INTRODUCE:**

Start by asking the class if anyone has heard of robotics. **What is a robot? Does a robot really “understand” what people say?** The answer to the last question is:













***“Not the same way that a person does.”***

Robots operate off of “instructions,” specific sets of things that they have been preprogrammed to do. In order to accomplish a task, a robot needs to have a series of instructions (sometimes called an algorithm) that it can run.

To get more familiar with the concept of an algorithm, it is helpful to have something to compare it to. For this exercise, we will introduce a programming language made of lines and arrows.

**PROGRAMMING KEY**


---

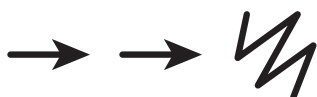
		Move One Square Forward
		Move One Square Backward
		Move One Square Up
		Move One Square Down
		Change to Next Color
		Fill-In Square with Color

---

In this instance, the symbols on the left are the “program” and the words on the right are the “algorithm” piece. This means that we could write the algorithm:

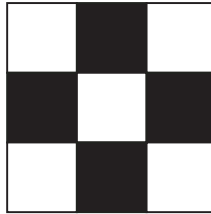
***“Move one square forward, Move one square forward, Fill-in square with color”***

and that would correspond to the program:



Now it's time to get a little more practice. Start your class off in the world of programming by drawing or projecting the provided key onto the board.

Select a simple drawing, such as this one to use as an example.

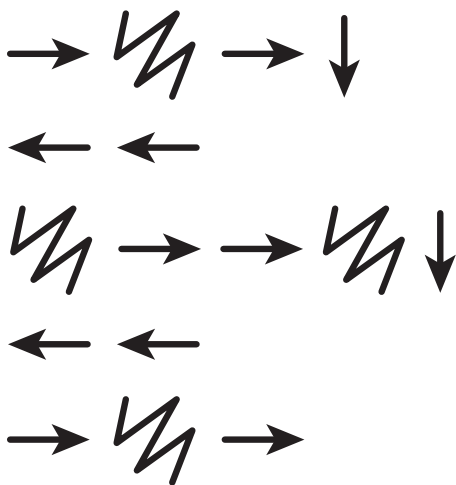


This is a good way to introduce all of the symbols in the key. To begin, you can show them the least confusing way to code an image, which is to return to the left of the image whenever you drop to the next line. Fill in the graph for the class, then ask them to help describe what you've just done. First, you can put it into words with an algorithm, then you can program what you've laid out.

#### A sample algorithm:

**“step forward, fill-in, step forward, next row,  
back, back,  
fill-in, step forward, step forward, fill-in, next row,  
back, back,  
step forward, fill-in, step forward”**

Some of your class may notice that there are some unnecessary steps, but hold them off until after the programming stage. Walk the class through programming the image:

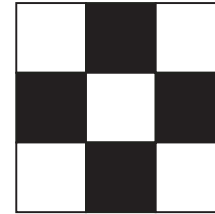


The classroom may be buzzing with suggestions by this point. If the class gets the point of the exercise, this may be a good place to entertain those.

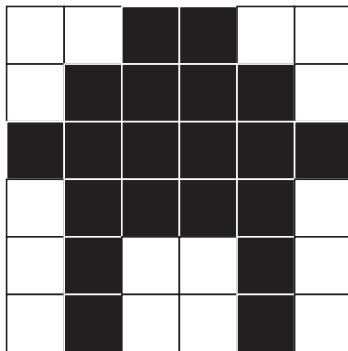
It is true that there is a bit of redundancy above, but it can be extremely confusing to code without it. Sometimes, until a programmer is experienced, it is helpful to code in the most understandable way to make sure it works, then remove unneeded steps later.

Work through the example again, first removing unneeded steps from the original program, then by coding from scratch without using any unnecessary symbols in the first place.

If the class seems lost or confused, try another graph but skip the “simplification” step. Just let it be right to left, then back again.



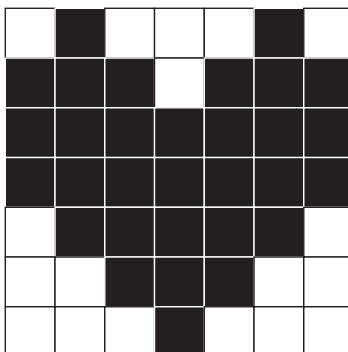
If the class can shout out the algorithm, then define the correct symbols to use for each step, they’re ready to move on. Depending on your class and their age, you can try doing a more complicated one together or skip straight to having them work in groups.



Give each group an image pack and suggest they choose one of the “mini one-color” images. Have them first write the algorithm, then convert it to symbols. Once they’ve done one or two, have them switch algorithms with another group and draw what the others programmed.

If there’s time, it is possible to introduce the need for functions and parameters. This is best accomplished with larger, more complicated drawings.

After doing a few of these, it becomes obvious that an image of this size is pretty hefty. Let’s look at just two lines from this one.



#### Algorithm:

**“Step forward, fill-in, step forward, step forward, step forward, fill-in, step forward, next line.  
Back, back, back, back, back, back.  
Fill-in, Step forward, fill-in, step forward, fill-in, step forward, step forward, fill-in, Step forward, fill-in, Step forward, fill-in, next line.  
Back, back, back, back, back, back.”**

Challenge the class to look for things that repeat often. Take some suggestions, but certainly one of the most helpful things to combine is the “Back, back, back, back, back, back.” Open the floor for suggestions of how you could turn this into a single symbol.

The class might arrive at something like this:

← 6

In fact, they may have already done something similar with the last batch of images. Are there any other combinations that can be made? What about skipping three squares in series? Coloring three squares in series? Coloring seven squares in series?

You are likely to have ended up with lots of symbols that contain various numbers. Hopefully by now the class understands what they are and why they're helpful.

Now comes the magic! You get to reveal that the class just discovered functions! They have created a simple representation for a complex grouping of actions. That is exactly what functions are meant to do. How about the number? It has a name also. That number is called a parameter. In the case of the example above, the parameter lets the function know how many times to move backward.

**Look at these “functions.” What do you suppose they do?**

1. ( → 6)
2. ( → ⚡ 6)
3. ( ⚡ → ↓ 6)

**Answers:**

1. Move forward six spaces
2. Color 6 blocks in a row
3. Color a diagonal line

Are there any other combinations that could be helpful?

Armed with this new method, challenge the class to choose one of the hardest images that they can handle. Do these symbols help the process go more quickly?

When the groups have finished their latest task, have them exchange their program (complete with functions and parameters) with another group and try to draw what the others have coded.

### ADJUSTMENTS:

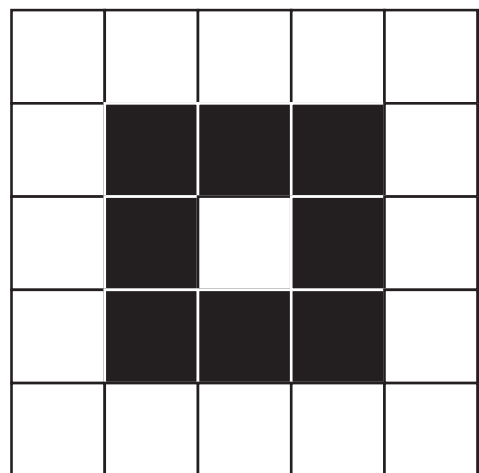
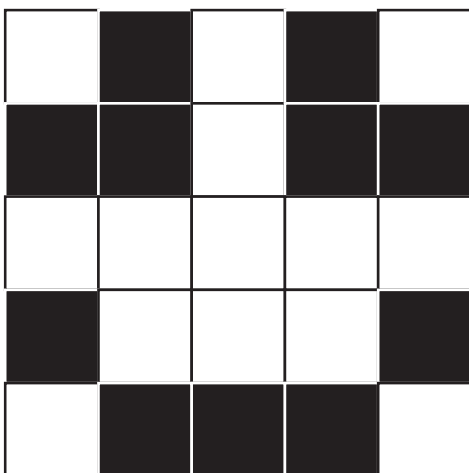
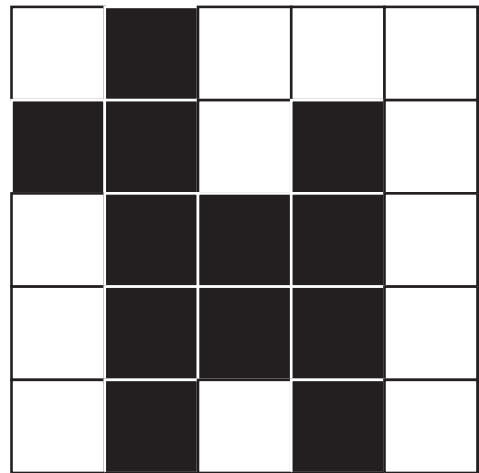
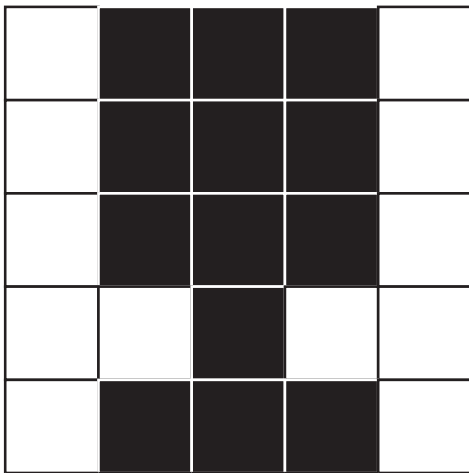
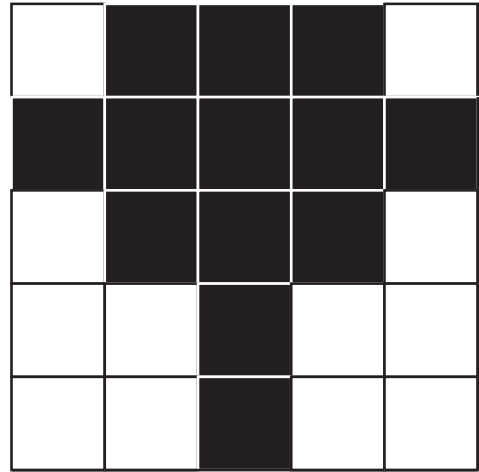
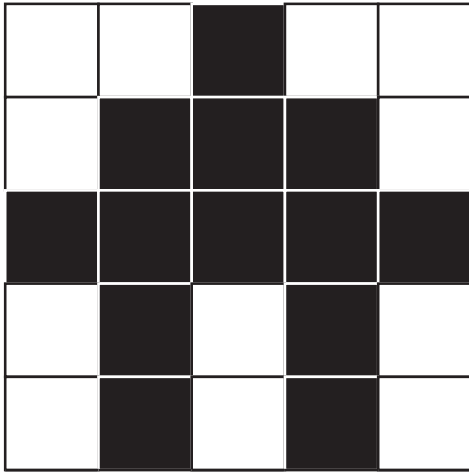
**K-2:** Have the class work together on a single image/algorithm. Use larger grid paper. Use single color images only.

**3-5:** Work in small groups (2-4). Use larger grid for shorter work times.

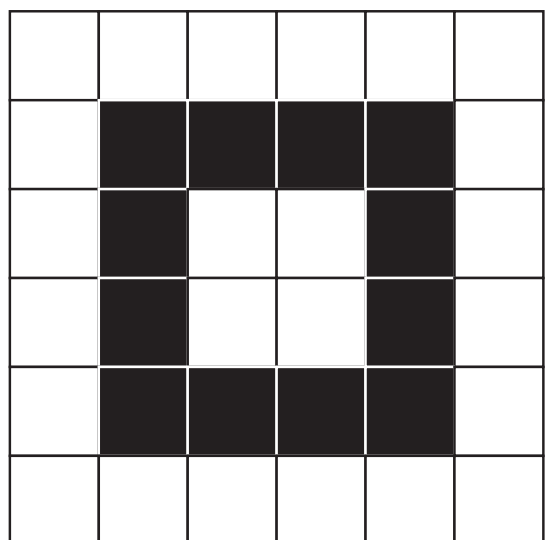
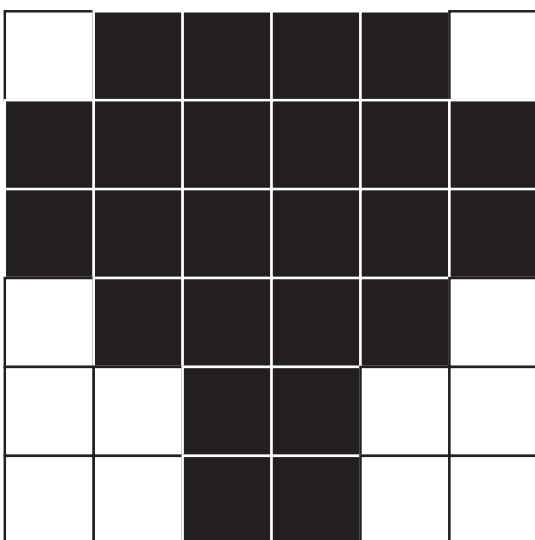
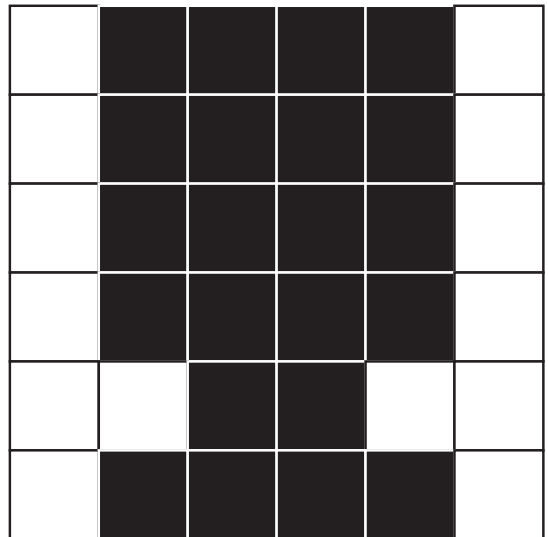
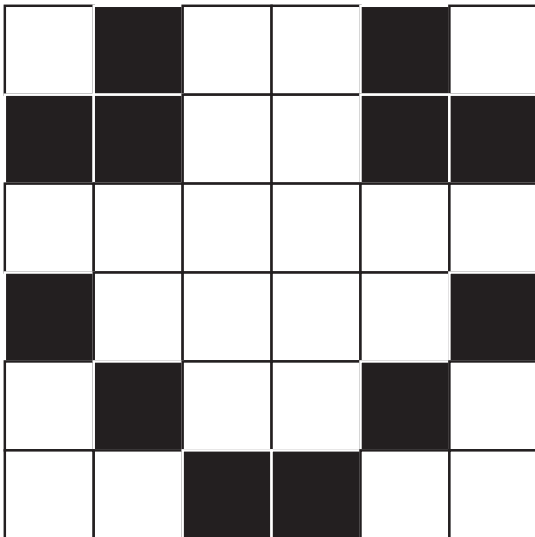
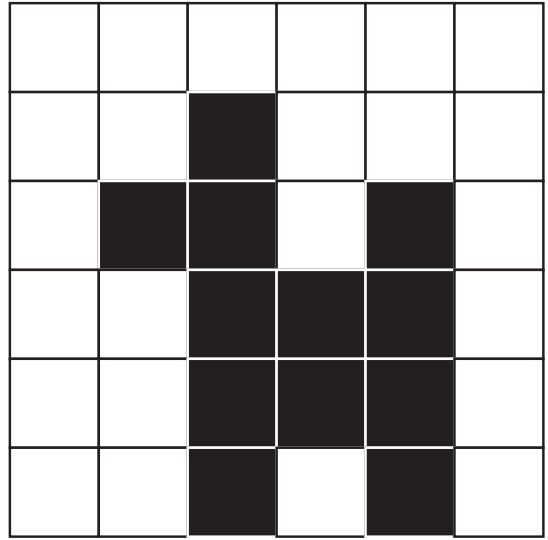
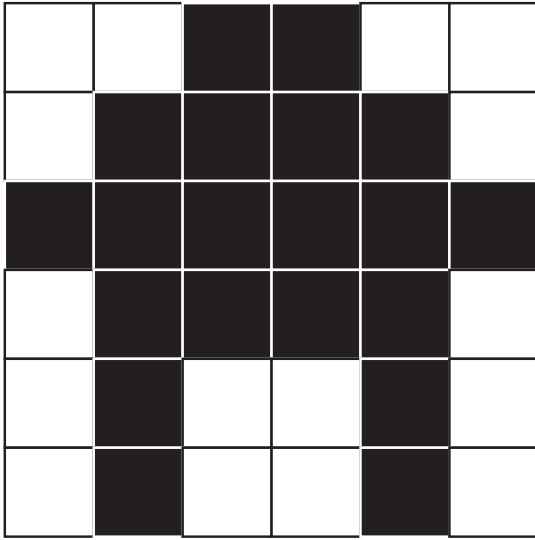
**6-8:** Work in pairs or as a solo activity

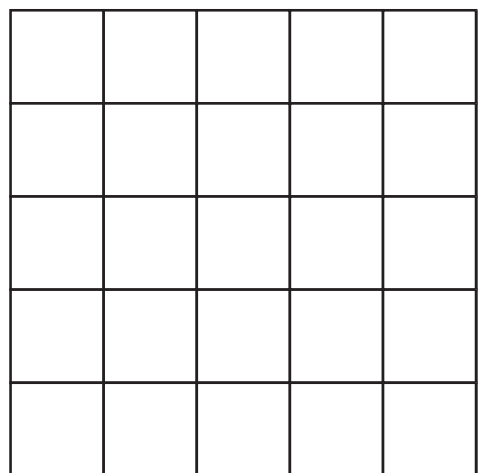
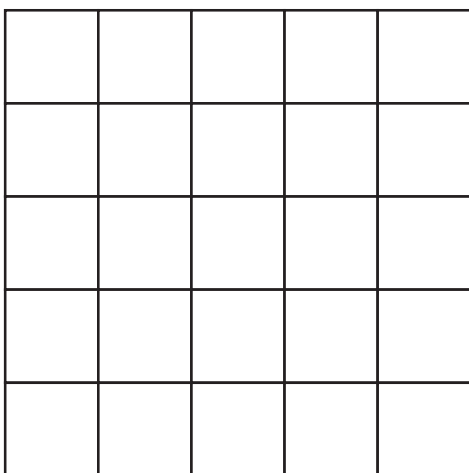
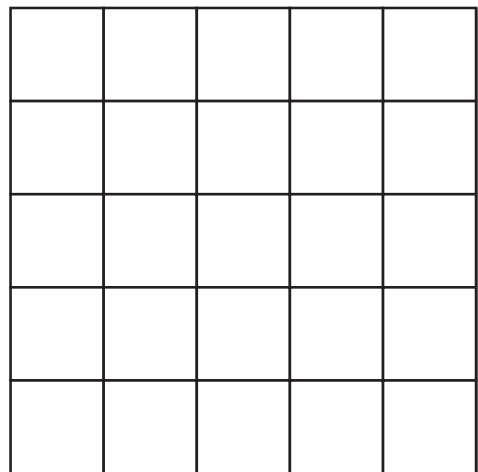
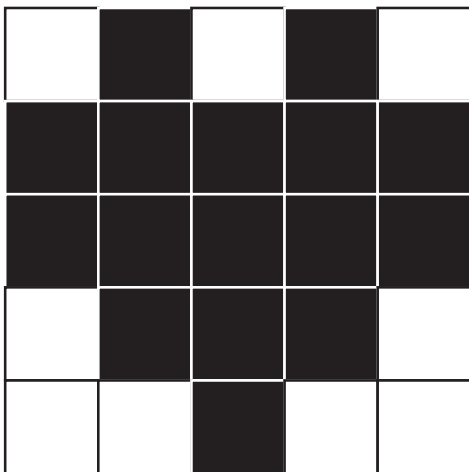
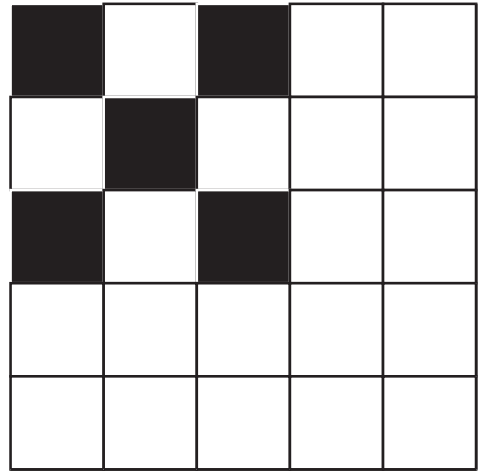
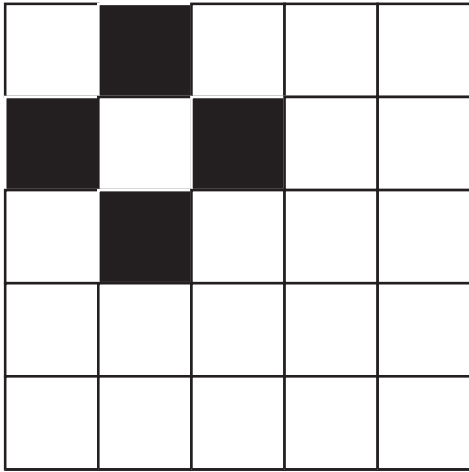
**STEPS:**

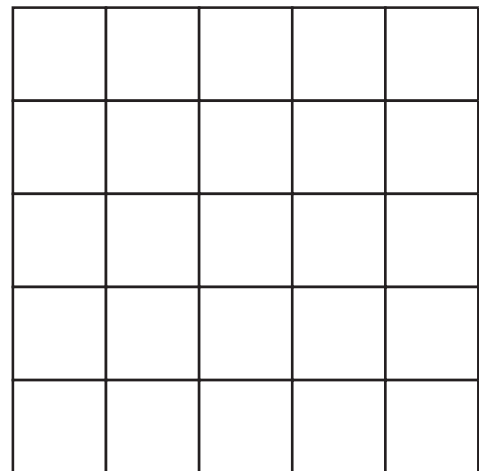
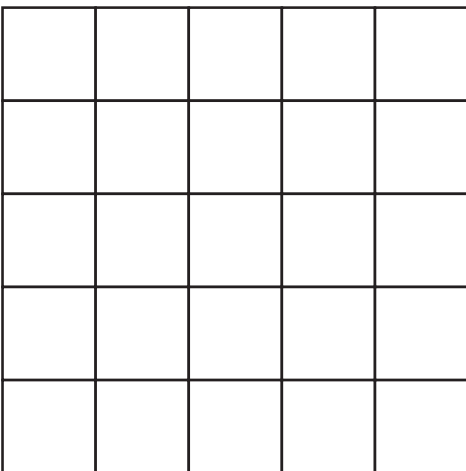
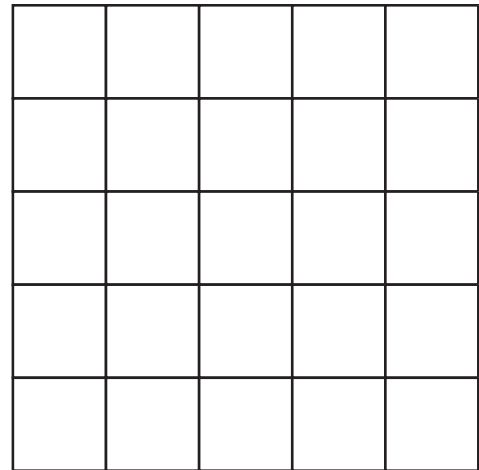
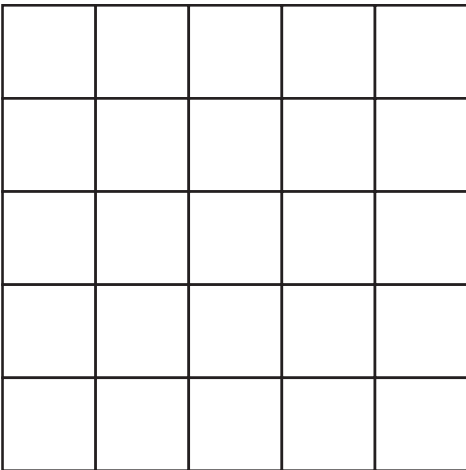
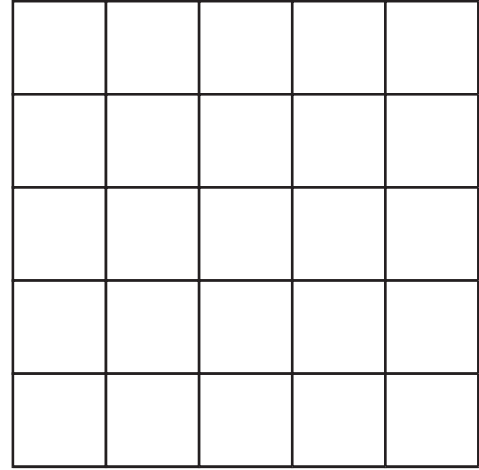
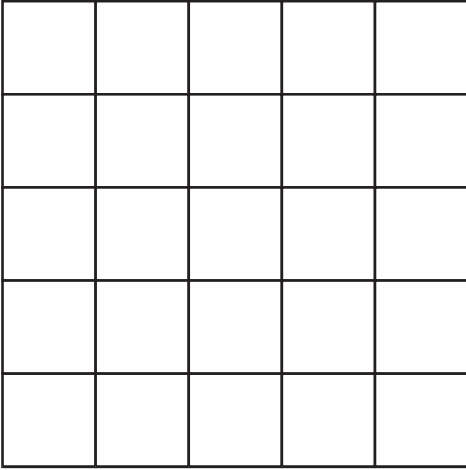
- 1) Choose image from image pack.
- 2) Write out algorithm to draw that image.
- 3) Convert algorithm into a program using symbols.
- 4) Trade programs with another team and draw their image.
- 5) Add “functions” to make programs more simple.
- 6) Write programs for more complex images.
- 7) Trade your complex programs and draw again.

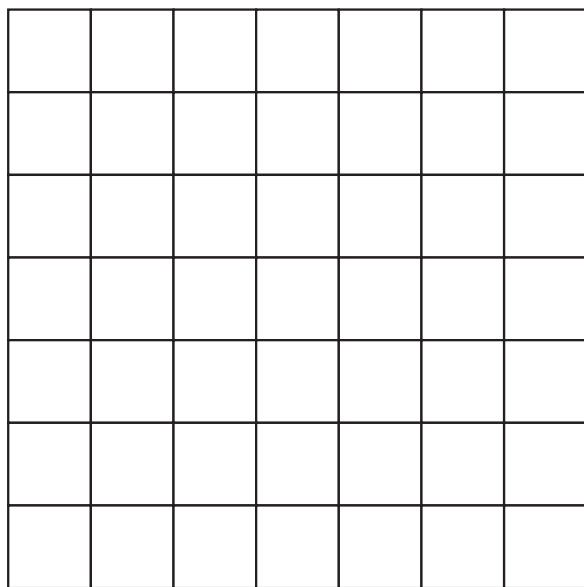
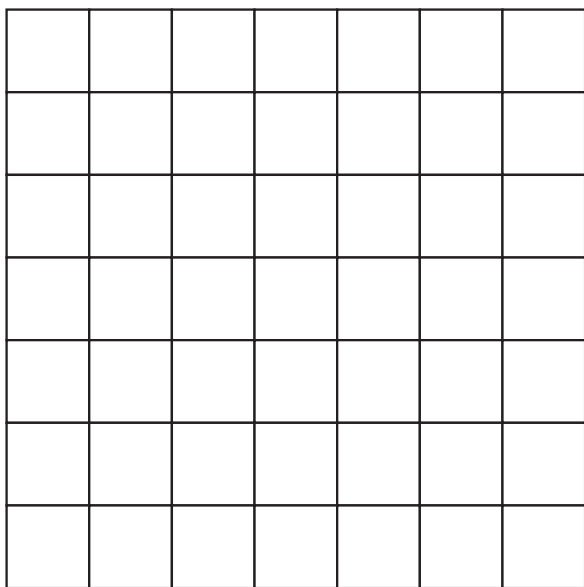
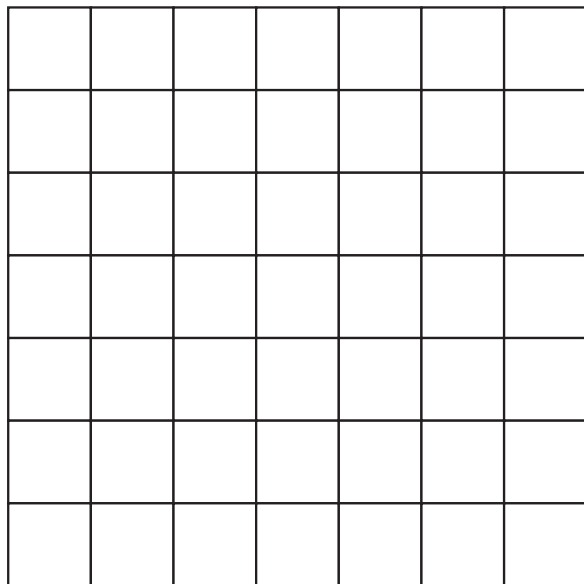
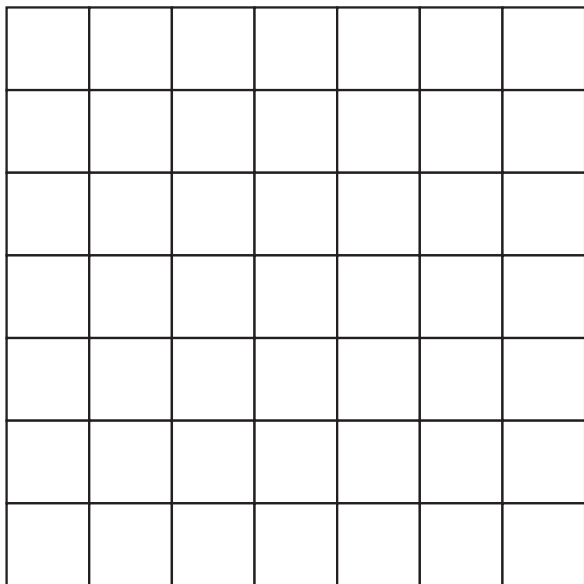


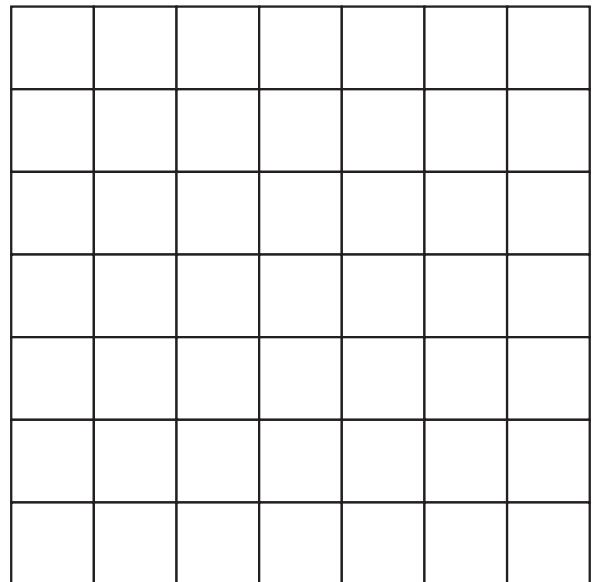
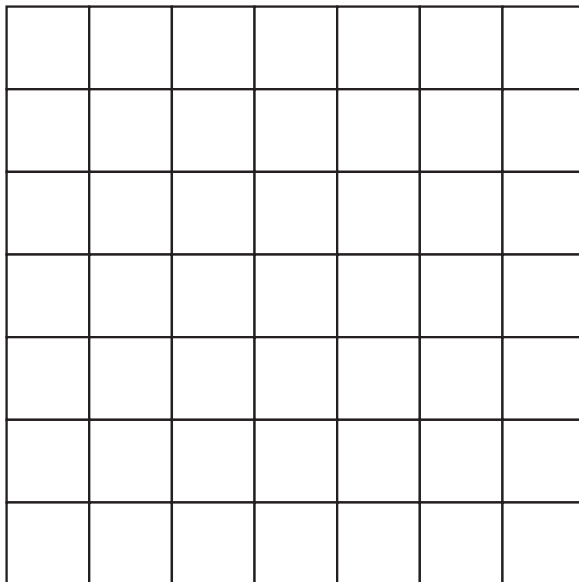
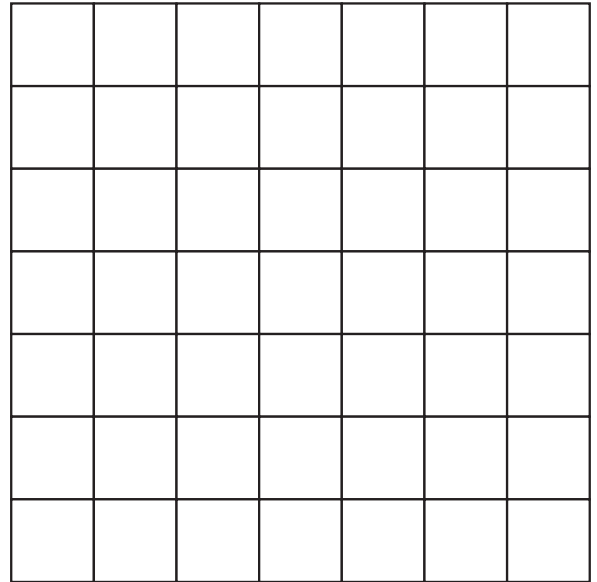
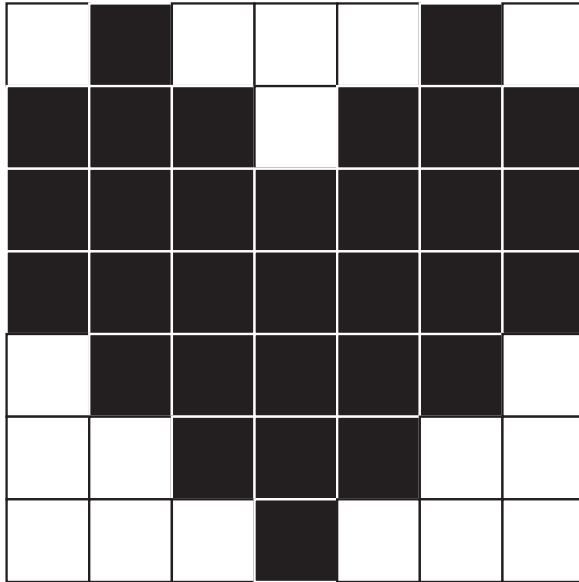




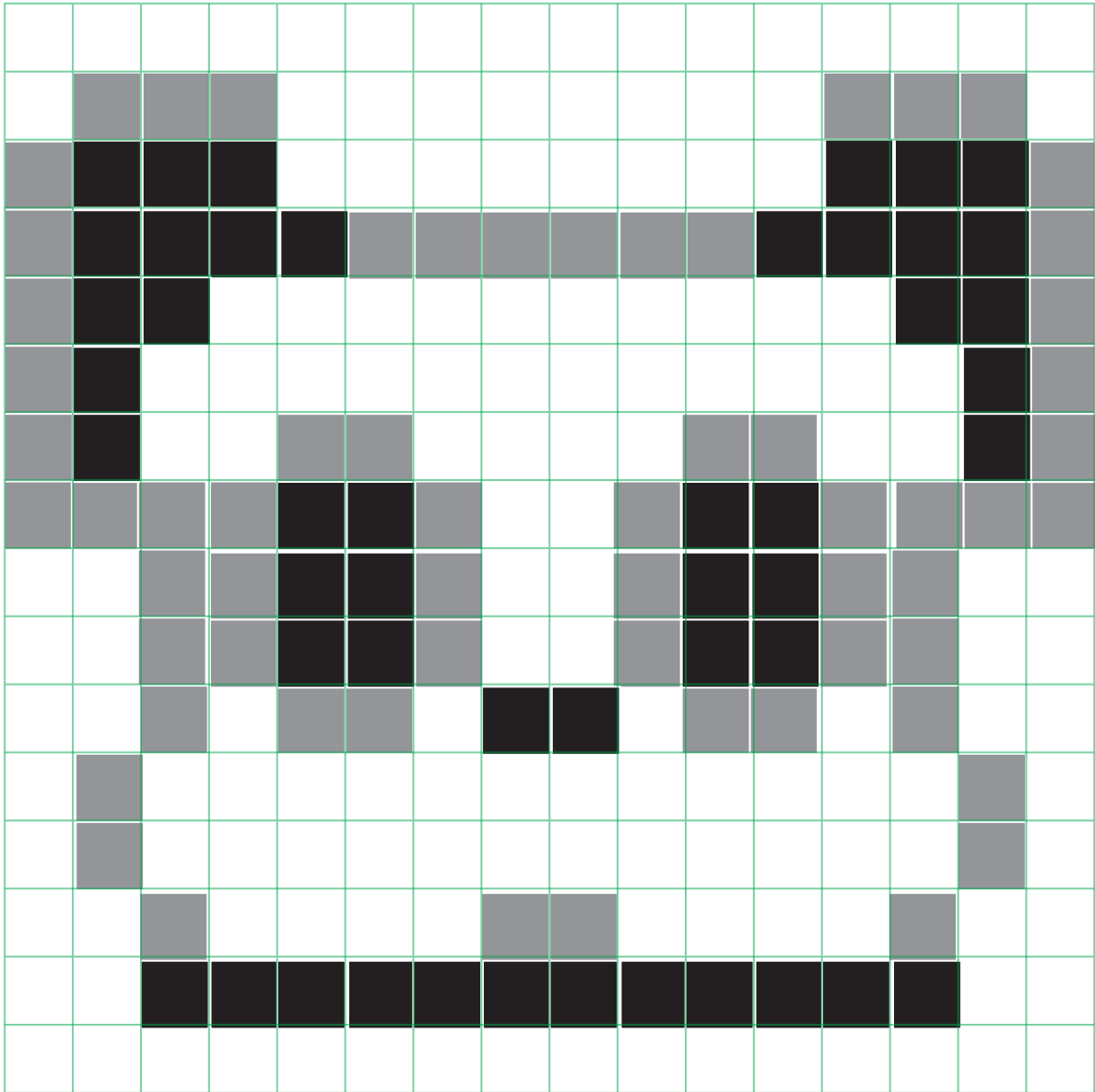
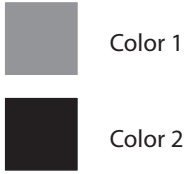








#### 4 LESSON 4: GRAPH PAPER PROGRAMMING



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:

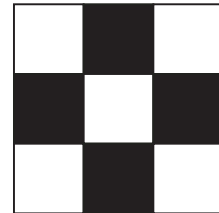
***“Mover un cuadrado adelante, mover un cuadrado adelante, pintar el cuadrado con el color seleccionado”***

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:



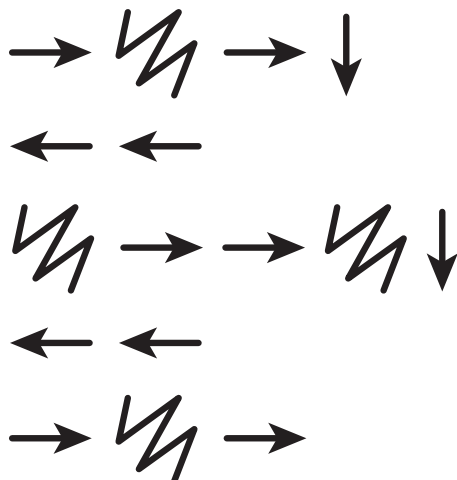
Esta es una buena manera de introducir todos los símbolos del lenguaje de programación. Para empezar, puedes mostrarles una forma menos confusa de codificar una imagen, que es retornar a la izquierda de la imagen siempre que terminas con la línea siguiente.

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:

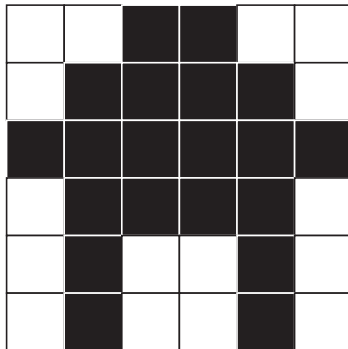
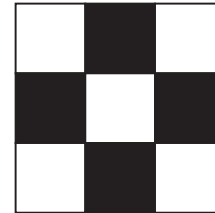


La clase puede realizar algunas preguntas a este punto. Si la clase entiende el objetivo de este ejercicio, entonces puede ser tiempo de entretenerlos.

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.

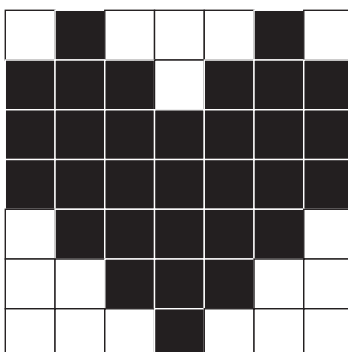


Si la clase puede descifrar el algoritmo, entonces define los símbolos a usar en cada paso, porque están listos para continuar con algo más complejo. Dependiendo de tu clase y la edad que tienen, puedes probar ejercicios más difíciles y permitirles trabajar en grupos.

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.

