Línea Ares — Metodología de Enseñanza de la Programación

ROBOTSchool · Documento maestro (fuente única) Versión 0.1 — Junio 2026

> Cómo se **enseña** a programar en Ares: el método y la **plantilla de lección** que seguirá toda la ruta de programación. Premisa: **programar es una habilidad de pensamiento con didáctica propia** — corre en una **ruta progresiva paralela** al proyecto y se **aplica** en él; no se aprende solo "dentro del robot". Las **definiciones aparecen en el flujo** (cuando el concepto se necesita), no en un glosario.

---

1. Por qué programar necesita su propia ruta (no como la robótica)

En robótica el proyecto puede ser el vehículo del aprendizaje. En programación, si el concepto vive solo dentro del proyecto, la carga de armar el robot **tapa** el concepto y el estudiante copia código sin entenderlo. Por eso la programación se enseña en una **secuencia deliberada de lecciones cortas** (la *ruta de programación*), y el proyecto es donde se **aplica y consolida**.

---

2. Los principios del método

1. **Definir en el flujo.** Cada término se presenta cuando se necesita, en tres tiempos: **definir** → **ejemplo** → **usar**. Nada de glosario aparte.

2. **De lo concreto a lo abstracto.** Desenchufado (sin computador) → representación (diagrama/pseudocódigo) → bloques → texto. El mismo concepto, subiendo de abstracción.
3. **Leer antes de escribir.** No se empieza de la página en blanco: primero se **lee, predice y corre** un programa que ya funciona; luego se **modifica**; al final se **crea**. (Enfoques **PRIMM** y **Usar→Modificar→Crear**, los más respaldados por la evidencia.)
4. **Practicar en micromundos.** Karel/Reeborg/Scratch para ejercitar el concepto **aislado**, sin el ruido del hardware.
5. **Construir el modelo mental de la máquina.** Enseñar a **trazar** (seguir el programa paso a paso, "hacer de computador") y a **depurar** como habilidad explícita.
6. **Espiral + práctica deliberada con retroalimentación.** Pocos conceptos, muchas veces, cada año más profundo; muchos ejercicios cortos con respuesta inmediata.
7. **El proyecto al final.** El robot da el sentido ("¿para qué?") y consolida; no es donde se ve el concepto por primera vez.

---

3. El corazón del método: leer → modificar → crear (PRIMM)

Toda lección hace que el estudiante **lea código que funciona antes de escribir el suyo**:

- **P — Predecir:** mira un programa corto y dice qué cree que hará.
- **R — Correr (Run):** lo ejecuta y compara con su predicción.
- **I — Investigar:** explora *por qué* funciona; se señala la parte nueva (el concepto).
- **M — Modificar:** cambia algo y observa el efecto.
- **M — Crear (Make):** escribe el suyo desde lo aprendido.

Esto baja el miedo a la página en blanco y enfoca la atención en el concepto.

---

4. Anatomía de una lección de programación (la plantilla)

Toda lección de la ruta sigue **estos 12 momentos** (cortos; una lección puede durar 1–2 clases):

#	Momento	Qué pasa
1	**Engancha** (desenchufado)	Juego/actividad sin computador que hace vivir el concepto
2	**Define (en el flujo)**	"Un **[término]** es … (1 frase)." + un ejemplo cotidiano
3	**Observa** (Predecir + Correr)	Se muestra un programa corto que **ya funciona**; el estudiante predice y lo corre
4	**Investiga**	Preguntas sobre por qué funciona; se señala la parte nueva
5	**Representa**	Dibuja el **diagrama de flujo** o el **pseudocódigo** del ejemplo
6	**Modifica**	Cambia algo del programa y observa el efecto
7	**Practica** (micromundo)	Reto en Karel/Reeborg (o Scratch) con el concepto aislado + un **Parsons** (ordenar líneas)
8	**Traza**	Sigue un programa paso a paso a mano (modelo mental de la máquina)
9	**Crea**	Escribe su propio programa (con reto y solución)
10	**Depura**	Un programa con un **error** para encontrar y corregir
11	**Aplica**	Cómo este concepto se usa en el **proyecto del libro**
12	**Evalúa + gamifica**	

#	Momento	Qué pasa
		Hito (“es capaz de…”), indicador de rúbrica e **insignia**

> No toda lección usa los 12 momentos con el mismo peso (en los pequeños pesa 1, 2, 7; en los grandes 3–10), pero el **orden** y la lógica se mantienen.

---

5. Cómo se define un término “en el flujo” (la regla)

Cuando aparece un concepto nuevo, **siempre** en este patrón corto, dentro de la lección:

> **Bucle.** Un **bucle** es una instrucción que **repite** otras. *Ejemplo:* en vez de escribir “salta” diez veces, escribimos “repite 10 veces: salta”. *Ahora úsalo:* [reto].

Definición de **una frase** + **ejemplo cotidiano** + **uso inmediato**. La definición no se memoriza: se usa.

---

6. Secuencia de conceptos (en qué orden se introducen)

El orden de aparición de los conceptos a lo largo de la ruta (cada uno se **define en el flujo** la primera vez):

Banda	Conceptos que se introducen (en orden)
Exploradores	instrucción · **secuencia** · **evento** · depuración por intento
Constructores	**bucle** · **condicional** · operadores de comparación · **variable** · (asomada a **función**)
Inventores	**función** · parámetro · **lista** · datos · descomposición en módulos

Banda	Conceptos que se introducen (en orden)
Innovadores	estructuras de datos · **clase/objeto** · eventos de red · concurrencia

Cada concepto es una (o más) **lección(es)** con la plantilla de §4. La ruta es **espiral**: un concepto visto en una banda se repasa y profundiza en la siguiente.

---

7. Relación con los proyectos, los libros y la evaluación

- La **ruta de programación** (estas lecciones) corre **en paralelo**; el **proyecto del libro** es el momento 11 (“Aplica”) a gran escala.
- Cada lección emite **hito + indicador de rúbrica** (los de `19 §8`) e **insignia** (gamificación de `19 §10`).
- Las lecciones viven en la **plataforma** (interactivas, con PRIMM y micromundos) y en cuadernillos imprimibles.

---

8. Plantilla en blanco (para producir cada lección)

```
LECCIÓN [n] · [Concepto]   ·  Banda: [ ]  ·  Entorno: [Scratch/
1. Engancha (desenchufado): ____
2. Define: "Un [término] es ____." Ejemplo: ____
3. Observa (Predecir/Correr): [programa que funciona] → ¿qué ha
4. Investiga: [2 preguntas]
5. Representa: [diagrama / pseudocódigo]
6. Modifica: [qué cambiar y qué observar]
7. Practica: [reto en micromundo] + [Parsons]
8. Traza: [programa para seguir paso a paso]
9. Crea: [reto] → [solución]
10. Depura: [programa con bug] → [error]
11. Aplica: [cómo aparece en el proyecto del libro]
12. Evalúa + gamifica: hito ____ · rúbrica ____ · insignia ____
```

---

9. Pendiente

1. Producir la **ruta de lecciones** banda por banda con esta plantilla (empezando por Constructores).
2. Crear los programas de ejemplo (PRIMM) y los Parsons por lección.
3. Montar las lecciones interactivas en la plataforma.

Fin del documento v0.1. Es la base metodológica de `19` (didáctica) y del contenido por banda; se evalúa con la rúbrica de habilidad de `19 §8`.