# Flujo



**¡Comencemos!**

## ¿Qué aprenderás?

- Entender en qué consiste la programación.
- Conocer la definición de algoritmo y su importancia dentro de la programación.
- Conocer los elementos que forman parte de un diagrama de flujo
- Conocer la relación entre pseudocódigo y diagramas de flujo.
- Conocer la importancia de la lógica independiente del lenguaje.
- Conocer la importancia del desarrollo del pensamiento lógico.

## Introducción

### El computador como una calculadora inteligente

Pensemos el computador como la versión inteligente de una calculadora. Nosotros le ingresamos datos, ya sea por el teclado, desde un archivo, o desde una página web, el computador los procesa de acuerdo a un conjunto de instrucciones que especificamos y finalmente nos entrega resultados de acuerdo a esa especificación.

### ¿Qué es programar?

Programar consiste en escribir estas especificaciones paso a paso. Por ejemplo un programa básico podría ser:

```
Querido computador:
- lee un dato del teclado y guárdalo en el espacio1
- lee un dato del teclado y guárdalo en el espacio2
- suma el dato del espacio1 con el del espacio2 y muestra el resultado
al usuario
```

### Traduciendo las instrucciones

El computador no lee los programas escritos en este lenguaje directamente. Lo que se hace es ocupar programas compiladores o interpretadores que ponen las instrucciones en términos que pueda entender el computador.

## ¿Qué es un lenguaje de programación?

Un lenguaje de programación es un lenguaje formal que define un conjunto de reglas para escribir instrucciones para un computador.

Existen muchos lenguajes de programación, cada uno con sus ventajas y desventajas para diversos propósitos. Cada lenguaje tiene reglas propias.

### En Ruby

Por ejemplo nuestro programa anterior en Ruby se escribiría:

```
dato1 = gets.to_i
dato2 = gets.to_i
print dato1 + dato2
```

Programar es mucho más que conocer estas reglas.

## El trabajo de un programador

El trabajo de un programador no solo consiste en escribir estos códigos, sino también en entender y poder definir cuál es la serie de instrucciones que nos lleva al resultado deseado. Esto es lo que se conoce como algoritmo.

**¡Vamos con todo!**

## ¿Qué es un algoritmo?

Un algoritmo es una serie finita de pasos para resolver un problema.

Imagen 1. Origami.
Fuente: Desafío Latam

**Por ejemplo**: Para ensamblar un mueble, es necesario seguir todos los pasos del manual de forma secuencial.

Otros ejemplos típicos son una receta de cocina o un modelo de Origami.

### Una serie de pasos para resolver un problema

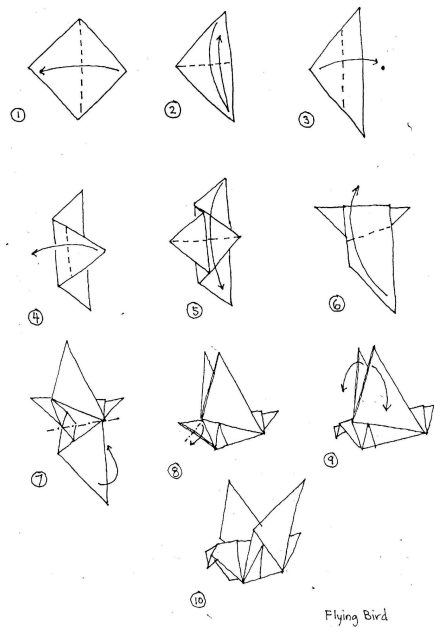


Imagen 2. Pasos Origami.
Fuente: https://www.laurenstringer.com/

Para indicar cómo construir una figura Origami, debemos plasmar claramente cada uno de los pasos; no podemos saltar ningún doblez.

## Escribiendo un algoritmo

Escribir un algoritmo es similar a especificar los dobleces. Debemos indicarle a un computador (o a una persona), el paso a paso de cómo resolver el problema.

Al igual que en el origami, descubrir cuáles dobleces logran los resultados esperados no es trivial y requiere de estudio y práctica.

### Formas de escribir un algoritmo

- Representar en un diagrama de flujo.
- Escribir en pseudocódigo.
- Implementar directamente en un lenguaje de programación.

## Diagrama de flujo

Un diagrama de flujo es una representación gráfica de un algoritmo, ayuda a visualizarlo en casos complejos.


Imagen 3.
Fuente: Desafío Latam.

Los diagramas de flujo suelen leerse de arriba hacia abajo y de izquierda a derecha, pero es la dirección de las flechas la que nos indica hacia dónde fluye el proceso. Los pasos se leen secuencialmente, o sea el paso 3 sucede después del paso 2 y el paso 2 sólo se ejecuta después del paso 1.

## Símbolos de un diagrama de flujo

Los diagramas de flujo tienen varios símbolos, pero los 4 más utilizados son:

- Inicio y fin del procedimiento.
- Entrada y salida de datos.
- Procesos (instrucciones que ejecuta la máquina).
- Decisiones.

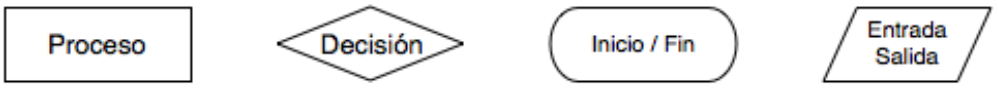


Imagen 4. Símbolos para diagramas.
Fuente: Desafío Latam

## Escribiendo nuestro programa como diagrama de flujo

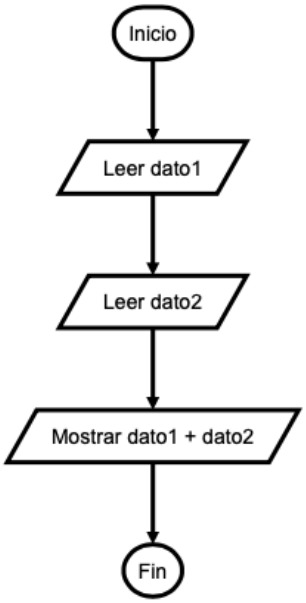


Imagen 5. Diagrama de Flujo.
Fuente: Desafío Latam

## Pseudocódigo

Es otra forma de representar un algoritmo, diseñada para la lectura. En pseudocódigo se utilizan palabras cotidianas, como la instrucción leer para especificar que el usuario tiene que ingresar un valor y mostrar para imprimir el valor en pantalla. Por ejemplo:

```
Algoritmo Suma
   Leer dato1
   Leer dato2
   Mostrar dato1 + dato2
FinAlgoritmo
```

## Diagramas de flujo y pseudocódigo

Todo diagrama de flujo puede ser transformado a pseudocódigo y viceversa. Todo pseudocódigo o diagrama de flujo puede ser implementado en un lenguaje de programación, aunque el traspaso no siempre será tan directo.

### Ventajas del pseudocódigo

El pseudocódigo, al igual que el diagrama de flujo, nos permite pensar en términos independientes al lenguaje de programación y concentrarnos en describir lo que estamos tratando de hacer y los pasos necesarios en lugar de cómo lograrlo.

En programación, es muy importante poder comunicar al computador lo que tiene que hacer paso a paso. Escribir pseudocódigo y diagramas de flujo ayuda a reforzar esta habilidad.

Otra ventaja es que son independientes del lenguaje, por lo que un algoritmo descrito en pseudocódigo puede ser traspasado a cualquier lenguaje de programación.

### Enfrentándose a un problema

En programación, al enfrentarnos a un problema, debemos abstraernos para:

1. Analizar el problema.
2. Descomponer el problema en partes que sepamos resolver.

3. Resolver cada parte del problema.

El pensamiento lógico corresponde a una competencia imprescindible al momento de programar y está directamente relacionada con nuestra capacidad para solucionar problemas.

## La importancia de desarrollar el pensamiento lógico

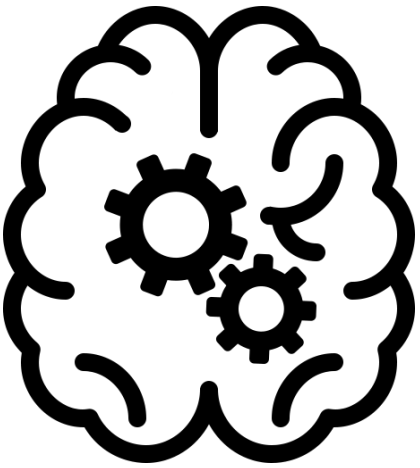

Imagen 6. Pensamiento lógico.
Fuente: Desafío Latam

Cuanto más desarrollemos nuestro pensamiento lógico, más rápido obtendremos soluciones a nuestros problemas cotidianos en programación.

Para desarrollar estas habilidades es importante que nos demos el tiempo necesario para hacer los ejercicios antes de revisar las soluciones, es normal que al intentar resolver los ejercicios por primera vez demore tiempo. La tolerancia a la frustración y el pensamiento lógico son las competencias claves que te convertirán en un buen programador.