Módulo 1: Lógica de Programación

1. Introducción a la Programación

La programación es el proceso de dar instrucciones a una computadora para realizar tareas. Un programador diseña estas instrucciones en un lenguaje comprensible para la máquina.

Práctica: Escribir en una hoja una lista de pasos detallados para hacer un sándwich e intercambiarla con otra persona para probar su claridad.

2. Pensamiento Computacional

Forma de abordar problemas de manera lógica y estructurada. Incluye la identificación de patrones y la creación de algoritmos.

Práctica: Identificar tres tareas diarias con un patrón repetitivo (ejemplo: lavarse los dientes) y escribirlas como pasos numerados.

3. ¿Qué es un Algoritmo?

Un algoritmo es una secuencia de pasos ordenados para resolver un problema.

Práctica: Crear un algoritmo para preparar un café.

4. Análisis de Problemas

Antes de programar, se debe entender el problema, dividirlo en partes pequeñas y definir entradas y salidas.

Práctica: Explicar a una persona sin conocimientos de tecnología cómo enviar un correo electrónico, detallando los pasos.

5. Tipos de Datos y Conversión de Datos

Diferencias entre números, texto y valores booleanos. Importancia de la conversión de datos en la programación.

Práctica: Escribir un algoritmo que pida la edad del usuario, la convierta a texto y la muestre en pantalla.

6. Estructuras de Control

Permiten modificar el flujo de ejecución de un programa. Se dividen en condicionales (decisiones) y bucles (repeticiones).

- **Condicionales:**
  - `if...else`: Si la condición es verdadera, se ejecuta un bloque de código. Si no, se ejecuta otro bloque (else).
  - `else if`: Permite evaluar múltiples condiciones en secuencia.
- **Bucles:**
  - `while`: Repite un bloque de código mientras se cumpla una condición.
  - `for`: Repite un bloque de código un número específico de veces.
  - `do...while`: Ejecuta un bloque de código al menos una vez y luego repite mientras se cumpla una condición.

Práctica: Escribir un algoritmo que pida un número e indique si es par o impar.

7. Operadores Lógicos y Relacionales

Herramientas esenciales para evaluar condiciones en los programas.

- `>`, `<`, `>=`, `<=`, `==`, `!=`, `!==`, `===`
- `AND`, `OR`, `NOT`

Práctica: Escribir un algoritmo que evalúe si un número es positivo, negativo o cero.

8. Funciones y Modularidad

Las funciones permiten dividir un programa en partes reutilizables, mejorando la organización y legibilidad.

9. Bucles Anidados

Se usan para manejar estructuras repetitivas más complejas, como tablas y patrones.

Práctica: Escribir un algoritmo que dibuje un rectángulo de asteriscos con dos bucles anidados.

10. Listas y Arreglos Básicos

Conjuntos de datos almacenados en una estructura ordenada.

Práctica: Crear un algoritmo que almacene 5 nombres en una lista y los imprima.

11. Paradigmas de Programación

Diferentes estilos para abordar la resolución de problemas con código.

- **Estructurado**
- **Orientado a Objetos**
- **Funcional**

Práctica: Investigar y escribir una breve definición de cada paradigma con un ejemplo.

12. Pseudocódigo y Diagramas de Flujo

Representación textual y gráfica de algoritmos antes de implementarlos en un lenguaje de programación.

Práctica: Dibujar un diagrama de flujo para hacer una llamada telefónica y escribir su pseudocódigo.

13. Depuración y Errores Comunes

Identificación y solución de errores en un programa.

- Errores de sintaxis
- Errores lógicos
- Errores en tiempo de ejecución

Práctica: Escribir un algoritmo con errores intencionales y corregirlo.

14. Trabajo Práctico Final

Aplicación de los conocimientos adquiridos para desarrollar un algoritmo funcional.

Práctica:

- Crear un algoritmo que pida dos números, los sume y muestre el resultado.
- Representarlo con un diagrama de flujo.