INSTITUTO KHIPU

SEMESTRE I
SESIÓN 01

CARRERA PROFESIONAL

DESARROLLO DE SISTEMAS DE INFORMACION

FUNDAMENTOS DE PROGRAMACIÓN

TEMA:

INTRODUCCIÓN A LA LÓGICA DE PROGRAMACIÓN

1. PREGUNTA DE INICIO

¿Cómo le explicas a un robot (que no tiene cerebro) cómo preparar una taza de té?

Si le dices: "Pon la bolsita en el agua", el robot podría poner la bolsa de té sobre un charco de agua fría en el piso. Tú no fuiste específico. Debiste decir:

1. Tomar una taza vacía.
2. Hervir agua a 100°C.
3. Verter el agua en la taza.
4. Introducir la bolsa filtrante.

Esto es **Lógica de Programación.** La computadora es ese robot: es increíblemente rápida, pero increíblemente tonta. Hace exactamente lo que le dices, incluso si le dices que se destruya a sí misma. Hoy aprenderás a darle las órdenes correctas.

2. CONCEPTOS BÁSICOS: EL ADN DEL SOFTWARE

2.1. DATO VS. INFORMACIÓN

En sistemas, estos términos no son sinónimos.

- **Dato:** Es la materia prima sin procesar. Hechos aislados.
  - **Ejemplo:** "25", "Rojo", "Cusco". (Por sí solos no significan nada).
- **Información:** Es el dato procesado y con contexto que sirve para tomar decisiones.
  - **Ejemplo:** "La temperatura en Cusco es 25 grados, alerta de calor". (Ahora es útil).
- **Rol del Desarrollador:** Nosotros construimos "máquinas" (sistemas) que comen Datos y defecan Información.

2.2. ALGORITMO VS. PROGRAMA

- **Algoritmo:** Es la secuencia lógica de pasos finitos para resolver un problema. Es universal (no depende del lenguaje). Es la receta.
- **Programa:** Es el algoritmo traducido a un lenguaje que la máquina entiende (Java, C++, Python). Es el plato cocinado.

El Arquitecto y el Albañil

El **Algoritmo** es el plano del arquitecto. Define dónde va la puerta, qué tan fuerte es la columna y por dónde pasan los cables. El **Programa (Código)** son los ladrillos que pone el albañil siguiendo ese plano. Si el plano (Lógica) está mal y dice que la puerta va en el techo, el albañil (Programador) construirá una puerta en el techo. El código será "perfecto", pero la casa no servirá. **Primero sé arquitecto (Lógica), luego sé albañil (Código).**

El Costo del Error Lógico

El dato del "1:100" (la Regla de los 100) es un estándar de la industria. Aunque a menudo se cita junto con el famoso reporte del **NIST** sobre el impacto económico de los errores de software (que calculó un costo de $59.5 mil millones anuales para la economía de EE.UU.), la gráfica específica de la progresión del costo (1x en diseño vs 100x en mantenimiento) proviene originalmente de estudios del **IBM Systems Sciences Institute** y es ampliamente respaldada por autores como Barry Boehm.

NIST
National Institute of Standards and Technology

3. FASES DE RESOLUCIÓN DE PROBLEMAS

El novato se salta las fases 1 y 2 y va directo a la 3. El experto respeta el ciclo:

1. Análisis del Problema: Entender QUÉ se pide. (Entrada - Proceso - Salida).

2. Diseño del Algoritmo: Diseñar CÓMO se resuelve (Pseudocódigo o Diagrama de Flujo).

3. Codificación: Escribirlo en el lenguaje de programación.

4. Pruebas y Depuración: Verificar que funciona y arreglar errores (Bugs).

> "La frustración es parte del trabajo. Si tu código no funciona a la primera, bienvenido al club. Programar es el arte de equivocarse 99 veces para que la vez número 100 funcione de maravilla y cambie el mundo."

4. PREGUNTAS CON RESPUESTAS GUIADAS

1. ¿Por qué necesito escribir el algoritmo en papel (pseudocódigo) si puedo hacerlo en la PC?

Porque el papel te permite borrar y tachar la lógica sin preocuparte por si te faltó un punto y coma (sintaxis). El papel estructura tu mente. Cuando pasas a la PC, solo traduces.

2. ¿Qué es un "Bug"?

Es un error en el programa. Puede ser de Sintaxis (escribiste mal una instrucción, el programa no corre) o de Lógica (el programa corre, pero hace lo incorrecto, ej. sumar en vez de restar). Los errores de lógica son los más difíciles de encontrar.

3. ¿Todo problema tiene un solo algoritmo de solución?

No. Un problema puede tener infinitos algoritmos.

- ***Problema:*** Ir de la Plaza de Armas a San Jerónimo.
- ***Algoritmo A:*** Taxi (Rápido, caro).
- ***Algoritmo B:*** Bus "El Zorro" (Lento, barato). Como desarrollador, tu trabajo es encontrar el algoritmo más eficiente (menos pasos, menos memoria).

4. ¿Qué es una "Variable" en lógica?

Es una cajita en la memoria donde guardas un Dato. Le pones nombre a la caja (ej. Edad) y guardas el valor (ej. 20). Durante el programa, puedes sacar el 20 y meter un 21. El valor "varía", por eso se llama variable.

5. ¿La lógica de programación sirve solo para programar?

No. Te enseña a pensar estructuradamente en la vida. Aprendes a descomponer problemas grandes (ej. "Aprobar el ciclo") en problemas pequeños y manejables (ej. "Estudiar 30 min hoy"). Es una habilidad de vida.

5. APLICACIÓN PRÁCTICA DEL CONTENIDO

Caso: "El Cajero Automático de la Caja Cusco"

Contexto: Te piden diseñar la lógica para que un cajero entregue dinero.

Fase 1: Análisis

- Dato de Entrada: Saldo del cliente, Monto a retirar.
- Restricción: No puede sacar más de lo que tiene.

Fase 2: Diseño del Algoritmo (Lógica Pura)

1. INICIO

2. LEER SaldoDisponible (Dato).

3. LEER MontoSolicitado (Dato).

4. SI MontoSolicitado > SaldoDisponible **ENTONCES**:

- **MOSTRAR** "Saldo Insuficiente".

5. SINO:

- NuevoSaldo = SaldoDisponible - MontoSolicitado.
- **ENTREGAR** Dinero.
- **IMPRIMIR** Voucher (Información).

6. FIN

Lección: Si no hubieras puesto el paso 4 (la condición lógica), el cajero regalaría dinero a quien tiene saldo cero. Un error de lógica en sistemas financieros es catastrófico. El código solo obedeció a tu lógica (buena o mala).

Según el **Foro Económico Mundial (WEF) - Future of Jobs Report** y proyecciones de la **Comisión Europea**, para el 2025-2030, más del **70% de las empresas** consideran el **"Pensamiento Analítico"** y las **"Habilidades Tecnológicas"** como competencias core obligatorias. Además, la Unión Europea ha fijado como objetivo que al menos el **80% de la población** tenga habilidades digitales básicas para 2030, ya que la mayoría de empleos (desde agricultura hasta derecho) requerirán interactuar con lógica de sistemas.

WORLD ECONOMIC FORUM

6. CONCLUSIONES

1. Orden Mental: Programar es 80% pensar y 20% escribir. Si no puedes escribir la solución en una servilleta, no estás listo para escribirla en la computadora.

2. Precisión: Las computadoras son literales. La ambigüedad es el enemigo. Debes aprender a dar instrucciones exactas y secuenciales.

3. Transformación: Tu misión es crear sistemas que tomen Datos (basura digital) y los conviertan en Información (oro digital) mediante Algoritmos eficientes.

7. MAPA MENTAL