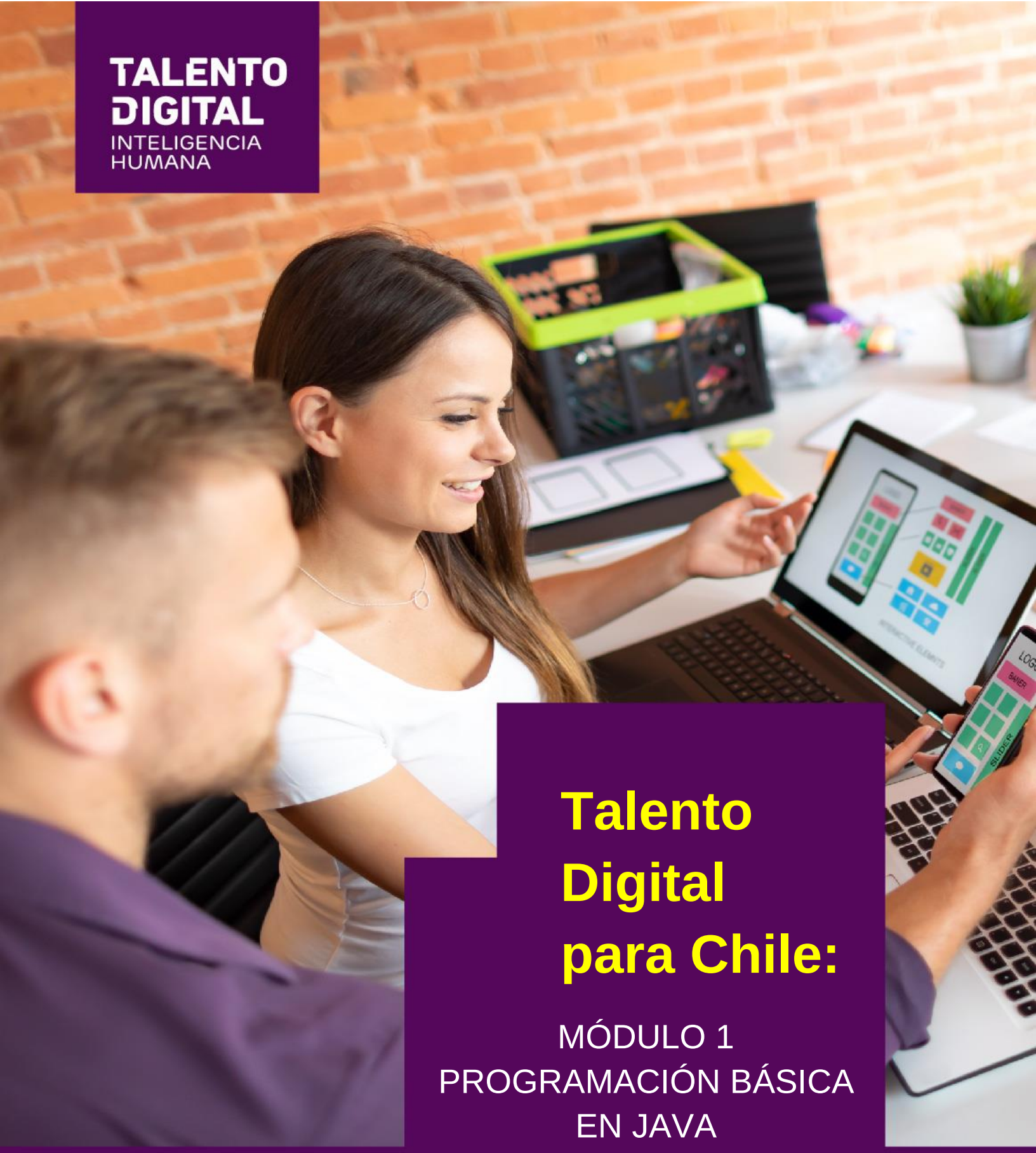
TALENTO DIGITAL
INTELIGENCIA HUMANA



Talento Digital para Chile:

MÓDULO 1
PROGRAMACIÓN BÁSICA EN JAVA

UN PROYECTO DE:

Ministerio de Hacienda · Ministerio del Trabajo y Previsión Social · CORFO · sence · SOFOFA · OTIC SOFOFA Capital Humano · CPC · ACTI · BID · FCH Fundación Chile · kodea

DESARROLLADO POR:

MÓDULO 1 - PROGRAMACIÓN BÁSICA EN JAVA

1.1 ALGORITMOS

Semana 1 - Día 1

Objetivo de la jornada

- Comprender el concepto de algoritmo.
- Desarrollar capacidades y mentalidad algorítmica para resolver problemas planteados según indicaciones entregadas.

Introducción a los algoritmos

A lo largo de la historia, el ser humano ha aprendido a resolver problemas, desde la creación de la rueda hasta la invención de los teléfonos móviles. En la actualidad, se siguen resolviendo problemas de forma innata hasta sin darse cuenta, sin embargo, no siempre se utiliza alguna metodología para llevarla a cabo, sino que se realiza basado en la práctica.

Para ejemplificar lo anterior basta con preguntarse, ¿Qué hacemos todas las mañanas al despertarnos? Seguramente la respuesta sería algo como esto:

- Pararse de la cama
- Ir al baño
- Arreglarse
- Desayunar
- Salir al trabajo/escuela

Al analizar la acción de “pararse de la cama”, se debe tomar en cuenta ¿Cuál sería el planteamiento del problema? Una respuesta acertada sería: “un día cualquiera por la mañana, una persona debe levantarse de la cama, pero esta se encuentra tapada por una sábana y dos cobijas”. Con el problema identificado, se pueden identificar los pasos a seguir para lograr levantarse de la cama. Este cuestionamiento se denomina lógica algorítmica.

Definición de un algoritmo

Un algoritmo es un conjunto de instrucciones o reglas definidas y no ambiguas, ordenadas y finitas que permite, típicamente, solucionar un problema, realizar un cómputo, procesar datos y llevar a cabo otras tareas o actividades. Dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución o salida.

Los algoritmos son independientes de los lenguajes de programación. En cada problema el algoritmo puede escribirse y luego ejecutarse en un lenguaje diferente de programación. El algoritmo es la infraestructura de cualquier solución, escrita luego en cualquier lenguaje de programación.

En la vida cotidiana, se emplean algoritmos frecuentemente para resolver problemas. Algunos ejemplos son los manuales de usuario, que muestran algoritmos para usar un aparato, o las instrucciones que recibe un trabajador de su patrón.

En términos de programación, un algoritmo es una secuencia de pasos lógicos que permiten solucionar un problema.

Un algoritmo debe cumplir además con las siguientes características:

- **Finito**: el algoritmo debe acabar tras un número finito de pasos. Es más, es casi fundamental que sea en un número razonable de pasos.

- **Bien definido**: el algoritmo debe definirse de forma precisa para cada paso, es decir, hay que evitar toda ambigüedad al definir cada paso. Puesto que el lenguaje humano es impreciso, los algoritmos se expresan mediante un lenguaje formal, ya sea matemático o de programación para un computador.

- **Entrada**: el algoritmo tendrá cero o más entradas, es decir, cantidades dadas antes de empezar el algoritmo. Estas cantidades pertenecen además a conjuntos especificados de objetos. Por ejemplo, pueden ser cadenas de caracteres, enteros, naturales, fraccionarios, etc. Se trata siempre de cantidades representativas del mundo real expresadas de tal forma que sean aptas para su interpretación por el computador.

- **Salida**: el algoritmo tiene una o más salidas, en relación con las entradas.

- **Efectividad**: se entiende por esto que una persona sea capaz de realizar el algoritmo de modo exacto y sin ayuda de una máquina en un lapso de tiempo finito.

Un algoritmo debe escribirse sin ceñirse a las reglas de un lenguaje. Existen varias formas para describir las operaciones de las que consta un algoritmo:

- **Descripción textual**: consiste en describir los pasos de forma narrativa.
- **Lista de operaciones**: es similar al textual, pero enumerando los pasos, utilizando variables, etc.

- **Diagrama de flujo**: son una representación gráfica en la que se utilizan cajas, rombos, flechas y otros símbolos para indicar los pasos del algoritmo.

- **Pseudocódigo**: se utilizan palabras clave para identificar las estructuras del algoritmo, como alternativas, repeticiones, etc.

Partes de un algoritmo

Para poder llevar a cabo un algoritmo, este debe obedecer a la estructura básica de un sistema, lo que permitirá dar una adecuada solución a un problema propuesto. Esta estructura debe contener lo siguiente:

- **Entrada (Input)**: corresponde al insumo, a los datos necesarios que requiere el proceso para ofrecer los resultados esperados.

- **Proceso**: pasos necesarios para obtener la solución del problema o la situación planteada.

- **Salida (Output)**: Resultados arrojados por el proceso como solución.

Para ejemplificar lo mencionado recientemente, tomemos un caso matemático, donde se nos solicita un algoritmo para:

Ejercicio.1 "**Calcular el área de un triángulo rectángulo"**

Solución:

INICIO
1. Identificar las medidas de la base y la altura
2. Multiplicar la base por la altura y dividir por 2 el resultado
3. Mostrar resultado

FIN

Tenemos los datos de **entrada**: base y altura, los cuales nos permitirán realizar el **proceso** de calcular el área de un triángulo rectángulo (base * altura / 2) para obtener como **resultado** el área solicitada.

Ejercicio 2:
"**Averiguar si un número es primo o no",** suponiendo que razonamos de la siguiente forma:

"Del análisis del hecho de que un número N es primo si sólo puede dividirse por sí mismo y por la unidad, un método que nos puede dar la solución sería dividir sucesivamente el número por 2, 3, 4..., etc. y, según el resultado, podríamos resolver el problema". Un diseño del mismo sería:

1. INICIO
2. Poner X igual a 2 (X = 2, X, variable que representa a los posibles divisores de N)
3. Dividir N por X (N/X)
4. Si el resultado es entero, entonces N no es primo, y saltar al punto 9 (en caso contrario continuar el proceso en el siguiente punto, 5)
5. Incrementar X en una unidad
6. Si X es menor que N saltar al punto 3 (en caso contrario continuar el proceso en el siguiente punto, 7)
7. Declarar N es primo;
8. Saltar al Fin (punto 10)
9. Declarar N no es primo
10. FIN

Como parte del diseño de algoritmo está la selección de uno que sea razonablemente aceptable, entre todos los muchos posibles que resuelven el mismo problema (el ejemplo que acabamos de dar es claramente mejorable, pues si N no era divisible por 2 no tiene mucho sentido volverse a preguntar si lo es por 4).

Durante el diseño es posible y aconsejable, realizar comparaciones entre algoritmos que resuelven el mismo problema. La bondad de un algoritmo puede medirse por dos factores:

- El tiempo que se necesita para ejecutarlo. Para tener una idea aproximada de ello, basta con saber el número de instrucciones de cada tipo necesarias para resolver el problema. - Los recursos que se necesitan para implantarlo.

Así, una vez diseñado un primer algoritmo, conviene realizar una evaluación del mismo, cuestión a veces nada banal y sobre la que volveremos en capítulos posteriores. Si se decide que éste no es eficiente será necesario o bien diseñar uno nuevo o bien optimizar el original. Optimizar un algoritmo consiste en introducir modificaciones en él, tendentes a disminuir el tiempo que necesita para resolver el problema o a reducir los recursos que utiliza. (En el ejemplo 2 el algoritmo se optimiza, si N se declara como primo cuando X supera a N/2).

Recursos Bibliográficos: http://robotica.uv.es/pub/Libro/PDFs/CAPI3.pdf