Clase grabada

Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

alkemy

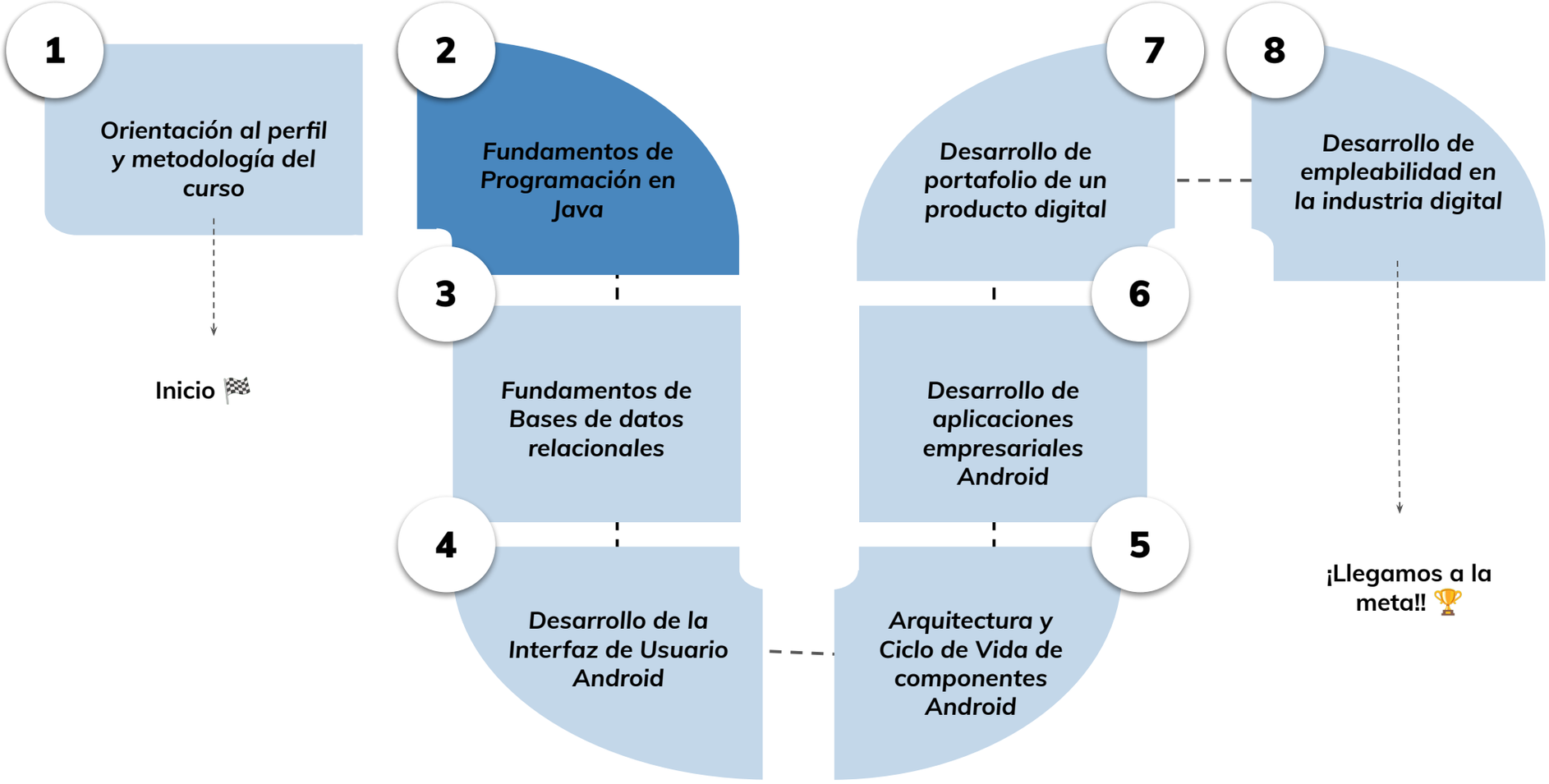
Clase grabada

Algoritmos

AE2.1: Estructurar un algoritmo lógico utilizando estructuras de control y expresiones para dar solución a un problema de baja complejidad acorde al lenguaje java.

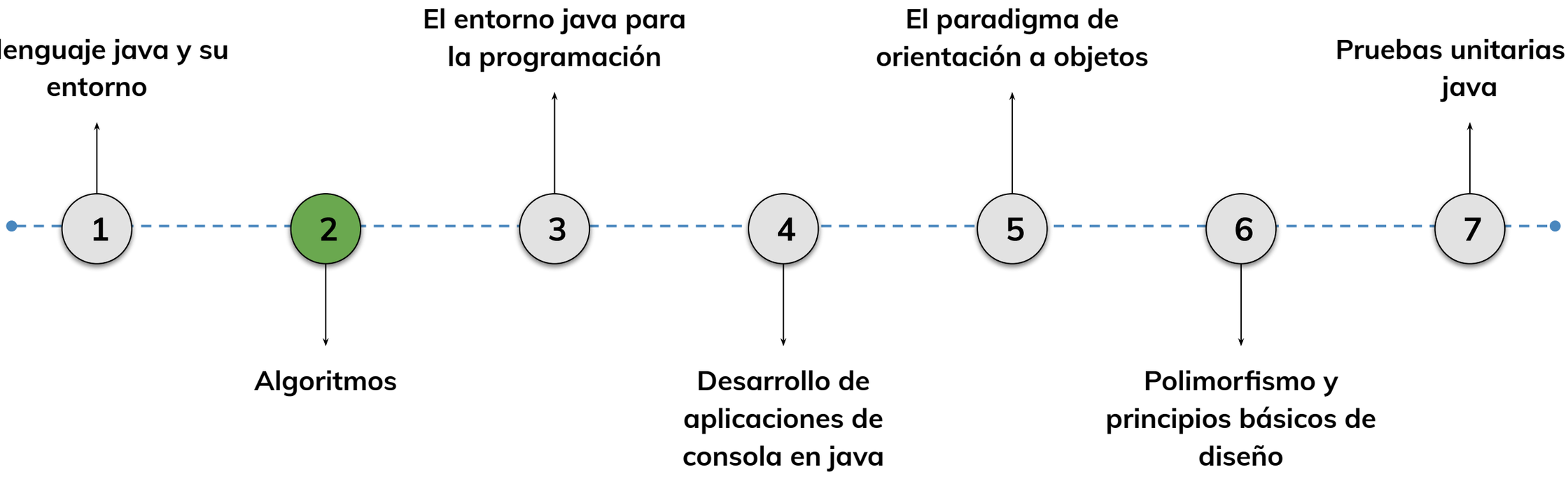
Hoja de ruta

*¿Cuáles **módulos** conforman el programa?*



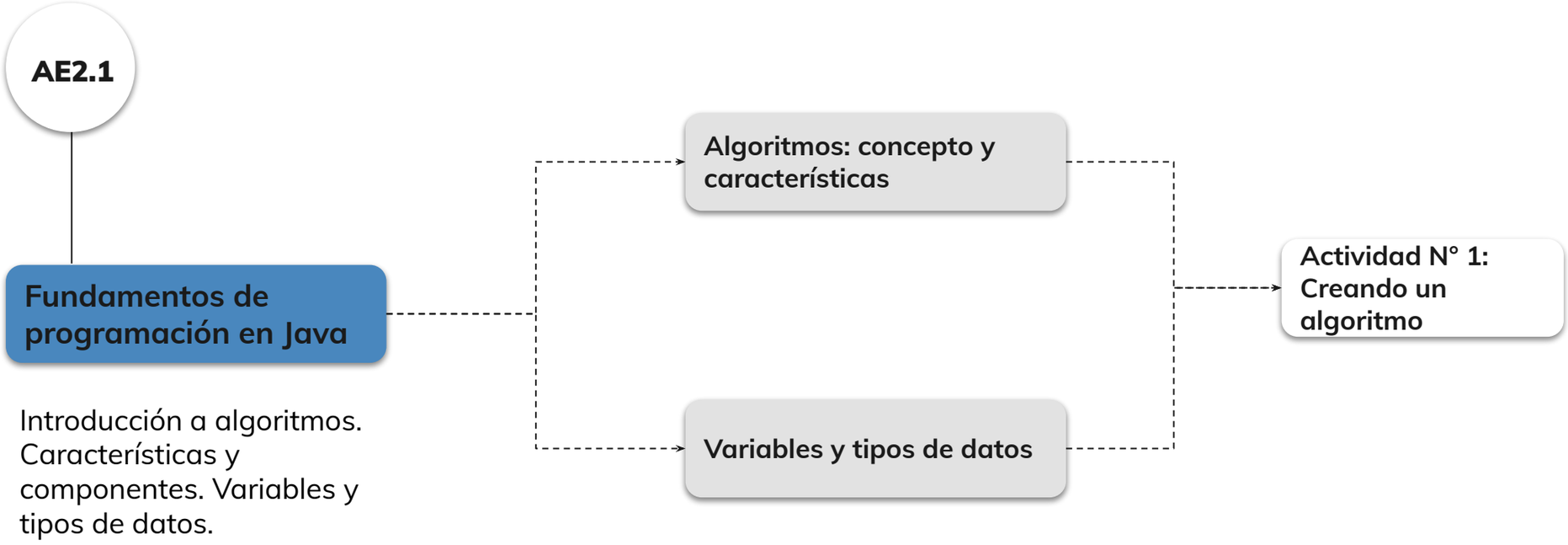
Roadmap de lecciones

*¿Cuáles **lecciones** estaremos estudiando en este módulo?*



Learning Path

¿Cuáles temas trabajaremos hoy?



alkemy

Objetivos de aprendizaje

¿Qué aprenderás?

- Comprender el concepto y las características de los algoritmos
- Entender el concepto y las características de las variables. Uso e implementación.
- Conocer los Tipos de Datos y las operaciones que pueden realizarse con cada uno.
- Comprender el uso de las expresiones aritméticas en un algoritmo.
- Conocer la utilización de los operadores aritméticos.
- Entender la implementación de los operadores relacionales.

Repaso clase anterior

¿Quedó alguna duda?

En la clase anterior trabajamos 📚:

- → *El lenguaje Java y sus características.*
- → *Los componentes de Java*
- → *Instalación del JDK y Eclipse IDE*
- → *Creación y ejecución de un proyecto Java.*

< > Rompehielo🥶

Compartamos experiencias: 🙌

¿Cómo explicarían qué es un algoritmo en sus propias palabras?

¿Pueden dar ejemplos de algún proceso paso a paso que siguen en su vida cotidiana?

- Les recomendamos abrir el mic 🎙️ o escribir por el chat😊

Algoritmos: concepto y características

¿Qué es un algoritmo?

Los algoritmos son una parte fundamental de la programación y la informática. Pero ¿qué es un algoritmo exactamente?

Un algoritmo es una secuencia finita de instrucciones bien definidas y no ambiguas, que permite realizar una tarea específica. Es como una receta o manual de pasos para resolver un problema particular. Por ejemplo, las instrucciones para instalar un programa en la computadora, las indicaciones para llegar a un lugar, o los pasos para hacer una tarta, son todos algoritmos.

En informática, los algoritmos nos permiten dar soluciones a problemas de procesamiento de información de manera precisa y automatizada. Son utilizados para el diseño de programas y software.

Partes de un algoritmo

Los algoritmos diseñados son independientes del lenguaje, es decir, son simplemente instrucciones que se pueden implementar en cualquier lenguaje y, sin embargo, el resultado será el mismo.

Partes de un algoritmo

- Entrada: son todos los datos que se introducen al algoritmo para que pueda ejecutarse en base a dicha información.

- Procesamiento: con lo que ha recibido en la entrada, el algoritmo llevará a cabo una serie de cálculos para dar con la solución al problema.

- Salida: los resultados que se han obtenido del procesamiento de datos se mostrarán en la salida del algoritmo.

Características de los algoritmos

Principales características:

- Secuenciales: se procesan consecutivamente. Responden a una secuencia de pasos determinados.
- Precisos: tienen que ser objetivos en la resolución del problema.
- Ordenados: deben leerse y ejecutarse en un orden específico.
- Finitos: deben contar con un número concreto de pasos.
- Concretos: deben mostrar una solución al problema especificado.
- Definidos: antes los mismos datos de entrada, siempre debe conseguirse los mismos datos de salida.

Variables

¿Qué es una variable?

Los datos que vamos a manipular en los algoritmos deben estar contenidos en una variable.

Las variables son contenedores que almacenan valores que pueden cambiar. Es un espacio en la memoria de la computadora donde se almacena temporalmente un dato durante la ejecución del programa.

A estas se les asigna un identificador o nombre, y el tipo de dato que van a almacenar.

El valor de las variables puede modificarse durante la ejecución.

Las constantes también almacenan valores pero su contenido no cambia durante la ejecución.

Características de las variables

Principales características:

- Deben comenzar con letra.
- No deben contener espacios en blanco.
- No deben contener meta caracteres.
- No se podrán utilizar palabras reservadas como nombres de variable.
- Los identificadores de las variables deben ser únicos e irrepetibles.

Al declarar variables, es recomendable asignar un valor inicial:

- Ayuda a evitar errores y resultados inesperados.
- Permite documentar el propósito de la variable según el valor inicial.

  Ejemplo: Entero cantidad = 0

Ejemplo declaración de variables

```
Inicio
    // Declaración de variables de tipo entero
    Definir edad, numeroEstudiantes como Entero

    // Declaración de variables de tipo real (decimal)
    Definir salario, promedio como Real

    // Declaración de variables de tipo texto (cadena)
    Definir nombre, direccion como Cadena

    // Declaración de variables de tipo booleano (verdadero o falso)
    Definir esEstudiante, esActivo como Booleano

    // Declaración de una variable de tipo carácter
    Definir inicial como Carácter
Fin
```

Variables locales y globales

Variables Locales:

- Se declaran dentro de un bloque de código como una función o bucle.
- Su ámbito está limitado a ese bloque, no existen fuera de él.
- Se crean y destruyen cada vez que se ejecuta el bloque de código.
- No pueden ser accedidas desde afuera del bloque que las contiene.
- Permiten reutilizar nombres de variables en diferentes partes.

Variables Globales:

- *Se declaran fuera de cualquier bloque, al inicio del programa.*
- *Su ámbito es toda la ejecución del programa, pueden usarse en cualquier parte.*
- *Se crean al comenzar el programa y se destruyen al finalizar.*
- *Pueden ser accedidas y modificadas desde cualquier parte del código.*
- *Deben tener nombres únicos en todo el programa.*

Tipos de Datos

¿Qué son los tipos de datos?

Un dato es cualquier elemento que necesitemos procesar en un programa. Para saber qué operaciones podemos realizar con estos datos, los clasificamos según su tipo.

Un tipo de datos es un conjunto de valores que tienen una característica en común y que responden a unas operaciones determinadas.

Dependiendo del lenguaje de programación se puede trabajar con unos tipos u otros. Por ejemplo hay lenguajes que distinguen entre números enteros y números decimales y otros en los que solamente se tiene el tipo de datos numérico, englobando tanto decimales como enteros en el mismo saco.

Tipos de Datos

En Java esta es la forma de declarar una variable con su respectivo tipo de dato

En todos los lenguajes de programación encontramos una clasificación de tipos de datos siempre presente, **los tipos de datos simples y los tipos de datos compuestos.**

Tipos de Datos

Simples:

También llamados tipos de datos primitivos, son aquellos que contienen un elemento único de un tipo de datos particular y no se pueden descomponer en varios datos independientes.

```
Ejemplos:
 -  Entero: 5
 -  Real: 5,5
 -  Caracter: ‘H’
 -  Booleano: Verdadero/Falso
```

Compuestos:

- También llamados tipos de datos complejos o estructurados, se componen de agrupaciones de tipos de datos simples.
- Ejemplos:
- Arreglos: es un conjunto de valores que se almacenan en una misma referencia de la memoria
- Objetos: podemos tener distintos tipos de datos simples, y además existen operaciones asociadas a esos objetos.

Dato String o “Cadena”

Definición:

En Java, un “dato string” es texto representado por la clase String (java.lang.String). Es un objeto inmutable que guarda una secuencia de caracteres

No existe un tipo primitivo string; se usa String (con S mayúscula). Si vas a modificar el texto muchas veces, usa *StringBuilder* (mutable) o *StringBuffer* (thread-safe).

Ejemplo

```
- public class DemoStrings {
-   public static void main(String[] args) {
-     String saludo = "Hola";
-     saludo.toUpperCase();            // String es
  inmutable
-     System.out.println(saludo);      // Hola
-     String mayus = saludo.toUpperCase();
-     System.out.println(mayus);       // HOLA
-
-     StringBuilder sb = new
  StringBuilder("Hola"); // mutable
-     sb.append(" ").append("mundo");
-     System.out.println(sb.toString()); // Hola
  mundo
-   }
- }
```

alkemy

LIVE CODING

Escribiendo un algoritmo:

Vamos a escribir un algoritmo que permita calcular el promedio de 3 calificaciones de un alumno.

1. *Definir variables*
2. *Definir tipos de datos*
3. *Leer datos de entrada*
4. *Realizar operaciones necesarias*
5. *Almacenar resultado*
6. *Mostrar resultado por pantalla*

Tiempo: 20 Minutos

LIVE CODING

Solución:

```
Inicio
    Definir calificacion1, calificacion2, calificacion3, promedio como Real

    Escribir "Introduce la primera calificación: "
    Leer calificacion1

    Escribir "Introduce la segunda calificación: "
    Leer calificacion2

    Escribir "Introduce la tercera calificación: "
    Leer calificacion3

    promedio <- (calificacion1 + calificacion2 + calificacion3) / 3

    Escribir "El promedio de las calificaciones es: ", promedio
Fin
```

Ejercicio N° 1

Creando un Algoritmo

Creando un Algoritmo

Contexto: 🙌

En salas reducidas, deberán crear un algoritmo que permita resolver una receta de cocina a su elección. Este algoritmo puede escribirse como texto en un archivo doc.

Luego, compartir con el resto del curso cuáles fueron las partes más difíciles de definir.

Consigna: ✍️

El algoritmo debe tener:

- Datos de entrada: identificar qué variables serán necesarias, y el tipo de dato que guardaran.
- Procesamiento: definir qué hacer con las variables.
- Salida: resultado esperado

Tiempo 🕐 **:** *25 minutos*

¡Atención! Recuerda que el nombre o identificador debe ser simple y descriptivo.

¿Alguna consulta?

alkemy

Momento:

Time-out!

⏰5 -10 min.

alkemy

Expresiones aritméticas

Expresiones aritméticas

Las expresiones aritméticas son similares a fórmulas matemáticas.

En el código de la computadora, los operadores y las funciones se ejecutan a través de conjuntos de expresiones que incluyen expresiones aritméticas, de caracteres y booleanas. Estos proporcionan la base para los tipos de trabajo de datos que se realizan dentro de un programa de software.

Existen diferentes tipos de expresiones aritméticas en la sintaxis de programación: enteros o números reales, y números de punto flotante (para identificar y almacenar números complejos que pueden no encajar en un valor entero).

Operadores aritméticos

Las operadores aritméticos son:

- + Suma
- – Resta
- * Multiplicación
- / División
- -,**,^ Exponenciación
- div División entera (cociente)
- mod Módulo (resto de una división)

Los operadores se utilizan de igual forma que en matemáticas. Por consiguiente:

A x B se escribe en algoritmo como A * B, y 1/4 x C como C/4.

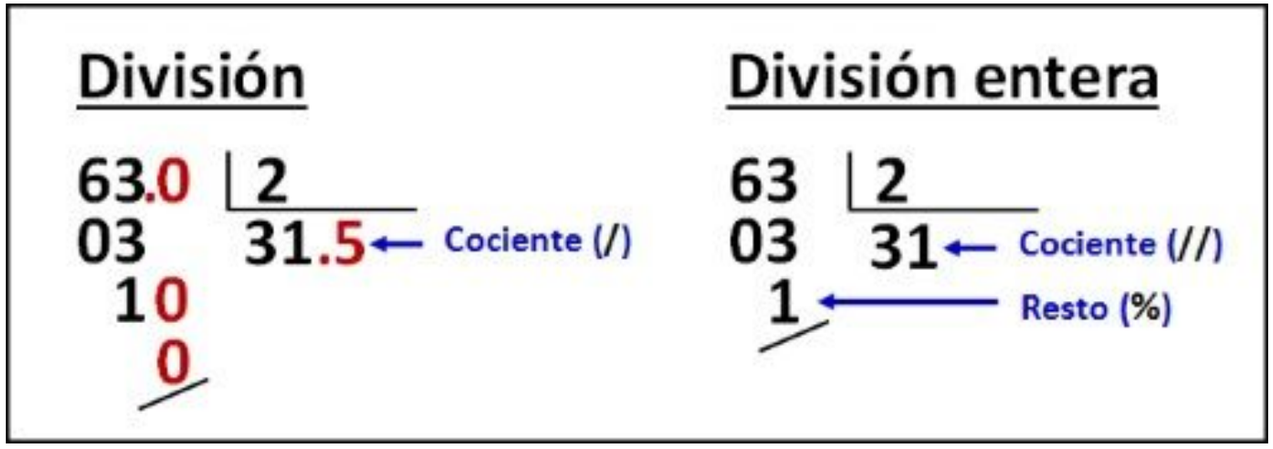
Al igual que en matemáticas, el signo menos juega un doble papel, como resta en A – B y como cambio de signo en – A.

Operadores aritméticos

Operador DIV: A div B = x

Obtiene la parte entera de A/B.

Operador MOD: A mod B = x
Obtiene el resto (módulo) de A/B



Operadores aritméticos

Reglas de prioridad

Las expresiones que tienen dos o más operandos, responden a reglas matemáticas que permiten determinar el orden de las operaciones, se denominan reglas de prioridad o precedencia.

Las operaciones que están encerradas entre paréntesis se evalúan primero. Si existen diferentes paréntesis animados (interiores unos a otros), las expresiones más internas se evalúan primero.

Operadores aritméticos

Reglas de prioridad

Las operaciones aritméticas dentro de una expresión suelen seguir el siguiente orden de prioridad:

- Operador exponencial (, o bien **).
- Operadores *, /, \,
- Operadores div y mod.
- Operadores +,-.

En caso de coincidir varios operadores de igual prioridad en una expresión o subexpresión encerrada entre paréntesis, el orden de prioridad en este caso es de izquierda a derecha.

LIVE CODING

Sacando cuentas:

Vamos a crear un algoritmo para realizar todas las operaciones aritméticas vistas hasta ahora. Para esto será necesario:

1. ***Entrada***: *2 variables numéricas*
2. ***Proceso:*** *operaciones aritméticas*
3. ***Salida:*** *1 variable numérica por cada operación.*

Tiempo: 15 Minutos

Ejercicio N° 2

Superliga

Superliga

¡A contar puntos!: 🙌

Las operaciones aritméticas son utilizadas a menudo en todo tipo de contextos. En este caso, vamos a aplicarlo al contexto de las competiciones deportivas. En salas reducidas, deberán crear un algoritmo que permita resolver el problema propuesto. Luego, compartir con el resto del curso cuáles fueron las partes más difíciles de definir.

Consigna: ✍️

Elaborar un algoritmo que permita ingresar el número de partidos ganados, perdidos y empatados por tu equipo favorito en la liga actual. Se debe mostrar el puntaje total, teniendo en cuenta que por cada partido ganado obtendrá 3 puntos, empatado 1 punto y perdido 0 puntos.

Tiempo🕐: *20 minutos*

Paso a paso: ⚙️

Los datos de entrada serán: cantidad de partidos ganados (pg), cantidad de partidos empatados (pe), cantidad de partidos perdidos.
También debes crear variables para calcular el puntaje por partidos ganados (ppg) y puntaje por partidos empatados (ppe). La suma de estas dos daría el puntaje total (ptotal).

Operadores Relacionales

Operadores Relacionales

Los operadores relacionales, también conocidos como operadores de comparación o lógicos, son símbolos que se utilizan en programación para comparar dos valores y obtener como resultado un valor lógico verdadero o falso.

Estos operadores son muy útiles en programación para comparar valores y expresiones, y para evaluar condiciones que dirigen la lógica de un programa. Se utilizan habitualmente para realizar validaciones.

Operadores Relacionales

Los principales operadores relacionales son:

- Mayor > : Compara si el valor de la izquierda es mayor que el de la derecha. Ejemplo: 7 > 5 devuelve true.
- Menor < : Comprueba si el valor de la izquierda es menor que el de la derecha. Ejemplo: 3 < 8 devuelve true.
- Mayor o igual >= : Verifica si el valor de la izquierda es mayor o igual al de la derecha. Ejemplo: 5 >= 5 devuelve true.
- Menor o igual <= : Revisa si el valor de la izquierda es menor o igual al de la derecha. Ejemplo: 5 <= 10 devuelve true.
- Igual == : Compara si dos valores son iguales. Ejemplo: 5 == 5 devuelve true.
- Distinto != : Compara si dos valores son diferentes.

  Ejemplo: 5 != 7 devuelve true.

Operadores relacionales	
>	Mayor.
<	Menor.
>=	Mayor o igual.
<=	Menor o igual.
==	Igual.
!=	Distinto.
=	Asignación.

Operadores Relacionales

Operador de asignación (=)

El operador (=) es un operador sobrecargado, es decir que dependiendo del contexto realiza distintas operaciones.

En el contexto de la asignación, el operador = se utiliza para modificar el valor de las variables, es decir, asignarle un nuevo valor. Ejemplo: x = 5.

Si lo que queremos es utilizar este operador como sinónimo de igualdad y no de asignación, se utiliza el operador dos veces: (==)

LIVE CODING

Practicando asignaciones:

Declararemos cuatro variables de **tipo entero *A, B, C y D*** *y le asignaremos un valor diferente a cada una. Vamos a utilizar sólo una variable auxiliar.*

A continuación, realizar las instrucciones necesarias para que:

- ***B*** *tome el valor de* ***C***
- ***C*** *tome el valor de* ***A***
- ***A*** *tome el valor de* ***D***
- ***D*** *tome el valor de* ***B****.*

Tiempo: 20 Minutos

alkemy

Ejercicio N° 3

Comparando

Comparando

Contexto: 🙌

Muchas veces es necesario comparar diferentes datos para resolver un problema. En este caso, vamos a evaluar diferentes operaciones y deberemos decidir qué valor lógico (verdadero o falso) responde a cada una.

Consigna: ✍️

Evaluar el valor de verdad de la siguiente expresión: 15 MOD 2=1

Evaluar el valor de verdad de la siguiente expresión: (10*10) <= (10^2)

Recuerda: ⚙️

- Reglas de prioridad de las expresiones.
- Los resultados sólo pueden ser 2: verdadero o falso

Tiempo 🕐 **:** *10 minutos*

¿Alguna consulta?

alkemy

Resumen

¿Qué logramos en esta clase?

- ✓ ***Conocer el concepto y las características de los algoritmos.***
- ✓ ***Identificar variables, qué son y cómo declararlas***
- ✓ ***Identificar los tipos de datos, qué son y qué operaciones se pueden realizar con cada uno.***
- ✓ ***Reconocer los operadores aritméticos y su implementación en programación.***
- ✓ ***Comprender los operadores relacionales y los valores lógicos en programación.***
- ✓ ***Identificar las diferencias entre comparación y asignación.***

¡Ponte a prueba!

Momento de ejercitación

Te invitamos a aprovechar esta última sección del espacio sincrónico para realizar de manera individual las ***actividades disponibles en la plataforma.*** *Estas propuestas son clave para afianzar lo trabajado y* ***forman parte obligatoria del recorrido de aprendizaje.***

💪 Análisis de caso ———— 💪 Selección Múltiple

💪 Comprensión lectora

Si al resolverlas surge alguna duda, compartela o tráela al próximo encuentro sincrónico.

alkemy

< ¡Muchas gracias! >