Operaciones aritméticas

¡Comencemos!

¿Qué aprenderás?

- Construir aplicaciones de tipo calculadora, donde el usuario ingresa valores y le entregamos resultados.

Para lograr esto tenemos que:

- Conocer los operadores aritméticos.
- Conocer la procedencia de los operadores aritméticos.
- Utilizar paréntesis para priorizar operaciones.
- Operar sobre números enteros y flotantes.

¡Vamos con todo!

Introducción a operaciones y operadores

En Ruby existen herramientas que nos permiten, entre otras cosas, sumar, restar, asignar valores, comparar, etc. Todo esto -y mucho más- es posible gracias a los operadores.

Motivación

¿Por qué debemos aprender operaciones matemáticas si estamos interesados en crear aplicaciones web o videojuegos?

Los operadores aritméticos los ocuparemos todo el tiempo, ya sea para calcular el total de un carro de compras o cambiar la posición de un personaje en un videojuego.

Operadores aritméticos

Los operadores aritméticos nos permiten realizar operaciones matemáticas sobre números.

Operador	Nombre	Ejemplo => Resultado
+	Suma	2 + 3 => 5
-	Resta	2 - 3 => -1
*	Multiplicación	3 * 4 => 12
/	División	12 / 4 => 3
**	Potencia	2 ** 4 => 16

Tabla 1. Operadores aritméticos.
Fuente: Desafío Latam

Operaciones con variables

El proceso es exactamente igual si guardamos los valores en variables

```
a = 2
b = 3
puts a + b # 5
```

Creando un calculadora

Esto nos permite que el usuario ingrese los valores, transformarlos a números y luego operar.

```
a = gets.to_i
b = gets.to_i
puts "a + b es: #{a + b}"
puts "a * b es: #{a * b}"
```

Este código podemos guardarlo en el archivo `calculadora1.rb` y ejecutarlo con ruby `calculadora1.rb`.

Precedencia de operadores

Un concepto muy importante que debemos conocer es el de precedencia. Dicho de otro modo, saber en qué orden se realiza un grupo de operaciones.

Por ejemplo, en la siguiente instrucción ¿cuál es el resultado?.

```
10 - 5 * 2

10 - 5 * 2 # 0
```

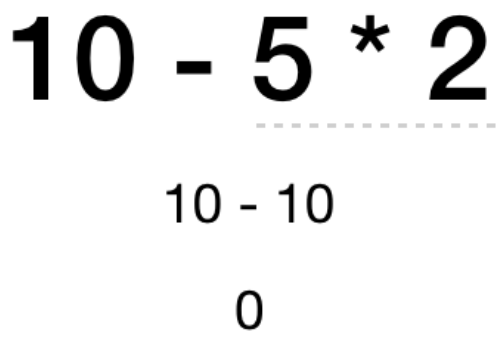

Imagen 1. Ejemplo de operación matemática.
Fuente: Desafío Latam

Orden de las operaciones

Veamos una tabla simplificada de precedencia. Esta tabla está ordenada de mayor a menor prioridad, esto quiere decir que la operación de exponenciación precede a (se realiza antes que) la suma.

Operador	Nombre
**	Exponenciación (potencia)
* , %	Multiplicación, división y módulo
+ , -	Suma y resta

Tabla 2. Operadores y priorización.
Fuente: Desafío Latam

Cuando dos operaciones tienen el mismo nivel de precedencia se resuelven de izquierda a derecha.

Operaciones y paréntesis

Al igual que en matemáticas, los paréntesis cambian el orden en que preceden las operaciones. Dando prioridad a las operaciones que estén dentro de los paréntesis.

```
(10 - 5) * 2
```

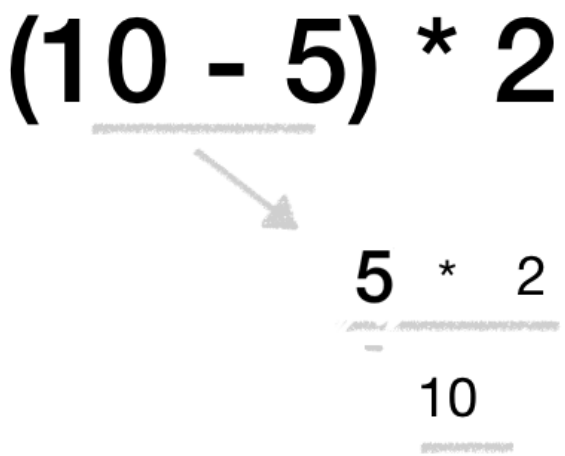

Imagen 2. Ejemplo de la importancia de los paréntesis.
Fuente: Desafío Latam

¡¡¡ Los paréntesis importan !!!

Operaciones con números enteros y decimales

Si dividimos números enteros nos encontraremos con una sorpresa.

```
5 / 3 # => 1
```

Esto es muy común, ocurre en casi todos los lenguajes de programación, pero normalmente esperamos una respuesta diferente. Para obtenerla, debemos ocupar otro tipo de dato, el float.

Float

En el capítulo anterior mencionamos que existía el tipo de dato asociado a los números decimales, llamado float.

```
(3.1).class # float
```

Enteros y floats

La división entre entero y float, o float y entero da como resultado un float.

```
5.0 / 3.0 # 1.6666666666666667
5 / 3.0 # 1.6666666666666667
```

La división entre floats también es un float.

Los floats son muy importantes dentro de la programación y tienen propiedades curiosas, una de las más importantes es que sólo almacenan una representación aproximada de los números.

```
(10 / 3.0)
 => 3.3333333333333335
```

También podemos transformar a float ocupando el método to_f, esto será especialmente útil cuando estemos trabajando con variables que contengan enteros.

```
a = gets.to_i # => 1
b = gets.to_i # => 2
puts a / b.to_f # 0.5
```

En algunos casos, también podemos transformar la variable a float desde un inicio.

```
a = gets.to_f
```

Ejercicios Resueltos

Realicemos los siguientes ejercicios

Pitágoras

La fórmula de Pitágoras nos permite calcular el largo de la hipotenusa de un triángulo rectángulo a partir de los largos de los catetos. Crearemos un programa donde el usuario introduzca los valores de ambos catetos y entreguemos como resultado el largo de la hipotenusa.

Para implementar el código sólo necesitamos conocer la fórmula.

$$h = \sqrt{c_1^2 + c_2^2}$$

Imagen 3. Fórmula de Pitágoras para determinar la hipotenusa.
Fuente: Desafío Latam

En Ruby se puede hacer el exponente como **, o sea se puede programar como a**2, también es lo mismo que a*a las dos opciones son válidas para resolver el problema

Algoritmo

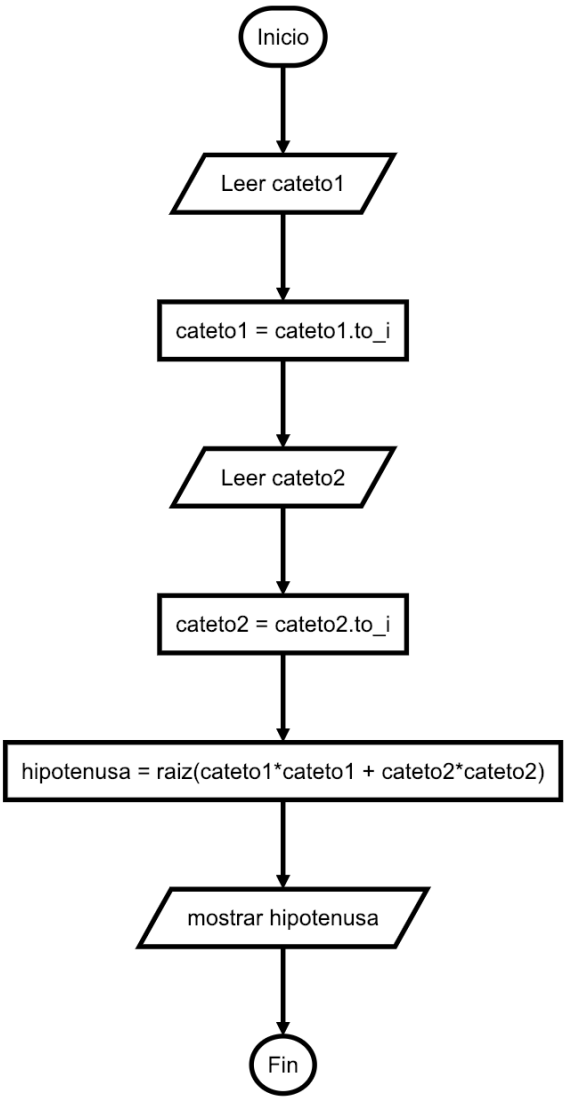

Imagen 4. Ejemplo de algoritmo con propiedades trigonométricas.
Fuente: Desafío Latam

Al leer un algoritmo tenemos que identificar qué partes no sabemos implementar, o pueden ser complejas. En el caso del algoritmo anterior lo único que no sabemos es calcular la raíz.

Si buscamos en Google square root y ruby encontraremos rápido la respuesta.

Para escribir la respuesta crearemos el archivo `pitagoras.rb` dentro de `introduccion-a-ruby/unidad-1` utilizando nuestro editor de código favorito.

Código

```
c1 = gets.to_i
c2 = gets.to_i
puts h = Math.sqrt(c1 ** 2 + c2 ** 2)
```

Probemos nuestro programa con `ruby pitagoras.rb` dentro de la carpeta `unidad-1`. Si ingresamos los valores 3 y 4 deberíamos obtener como respuesta 5

Math es un módulo, que en cierto modo es similar a una clase en el sentido de que tiene varios métodos. La documentación de un módulo se lee de igual forma que la de un objeto.

Fahrenheit

Fahrenheit y Celsius son dos escalas de temperatura frecuentemente utilizadas. Podemos transformar una temperatura de la escala Fahrenheit a Celsius ocupando la siguiente ecuación:

$$(\frac{f + 40}{1.8}) - 40 = c$$

Imagen 5. Fórmula para transformar los grados Fahrenheit en Celsius.
Fuente: Desafío Latam

Antes de desarrollar el algoritmo se deben revisar los siguientes puntos:

- ¿Cuántos valores debe ingresar el usuario? ¿1 o 2?.
- ¿Existe algún punto donde tengamos que tener cuidado con la precedencia de operadores?.

Luego dibuja el diagrama de flujo y finalmente escribe el código.

Algoritmo

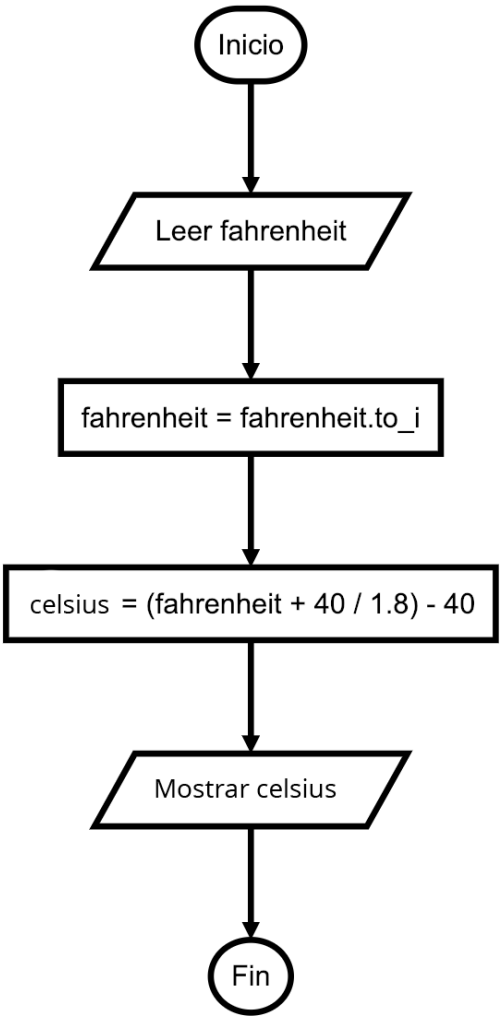

Imagen 6. Ejemplo de algoritmo con escalas de temperatura.
Fuente: Desafío Latam.

```
farenheit = gets.to_i
celsius = (fahrenheit + 40) / 1.8 - 40
puts "la temperatura es de #{celsius} celsius"
```

Ejercicio 1

Crea el programa `promedio3.rb` donde un usuario debe ingresar 3 valores y se debe calcular el promedio de estos. Recuerda guardar el programa en tu carpeta de estudio.

Ejercicio 2

Crea un programa `area.rb` donde el usuario ingresa el radio de una circunferencia y se debe calcular el área. Esta se calcula con la fórmula:


Recuerda guardar el programa en tu carpeta de estudio.

Resumen del capítulo

- **Precedencia:** El orden en que deben ser resueltas las operaciones.
- **División entre enteros:** El resultado es entero 6 / 4 #=> 0.
- **División entre flotante y entero:** El resultado es flotante 6 / 4.0 # => 1.5.
- **Paréntesis():** Al igual que en matemáticas nos ayudan a darle prioridad a una operación.