# Conceptos Básicos de Lógica

## Conceptos básicos de lógica

En tu proceso de aprendizaje, ten muy presente los siguientes conceptos para aplicarlos en los ejercicios de lógica de programación o desarrollo.

| CONCEPTO | EJEMPLO |
|---|---|
| **Template Strings**: son una forma más poderosa y flexible de trabajar con cadenas de texto y permiten la interpolación de variables, expresiones, y la inclusión de saltos de línea sin tener que usar concatenaciones complicadas con el operador +.<br>Estos en vez de estar dentro de las comillas simples convencionales están dentro de comillas invertidas también conocidas como “Backticks” que son representados por ` `. | `const nombre = 'Juan';`<br>`const saludo = `Hola, ${nombre}! ¿Cómo estás?`;`<br><br>`console.log(saludo); // Muestra "Hola, Juan! ¿Cómo estás?"` |
| **Función sin parámetros ni retorno**: una función es un bloque de código reutilizable que puede o no recibir parámetros y poder retornar algo o no retornar nada. En esté caso ejemplificamos una función vacía que no recibe ningún parámetro y tampoco nos da algún retorno | `function saludo() {`<br>`  console.log("somos Vankversity");`<br>`}`<br><br>`//Invocación de la función`<br>`saludo()` |
| **Función con parámetros:** una función con parámetros es una función que recibe uno o más valores de entrada (parámetros) al momento de invocarla. Estos valores pueden ser utilizados dentro del cuerpo de la función para realizar alguna operación o procesamiento. | `function sumar(a, b) {`<br>`  console.log("La suma de sus números es ", a+b);`<br><br>`}`<br><br>`sumar(5, 3);  // Se pasan 5 y 3 como argumentos` |
| **Funciones con parámetros con valores por defecto:** Estas son tipos de funciones las cuales dan un valor por defecto a uno o todos sus parámetros. Esto hace que al momento de ejecutarse si no se envía nada como argumento(si el parámetro correspondiente tiene un valor por defecto), la función usará un valor por defecto y se ejecutará la función con dicho valor. Los parámetros obligatorios(los que no tienen valor predeterminado) van primero, luego van los que tienen valores por defecto. | `function saludar(numero, nombre = "Amigo") {`<br>`  console.log(`${numero} Hola, ${nombre}!`);`<br>`}`<br><br>`saludar(1); // Salida: 1 Hola, Amigo!`<br>`saludar(2, "Juan"); // Salida: 2 Hola, Juan!` |

| | |
|---|---|
| **Función con parámetros y retorno:** una función con parámetros y retorno es una función que recibe uno o más valores (parámetros) como entrada y devuelve un resultado (valor de retorno) al finalizar su ejecución. | `function multiplicar(a, b) {`<br>`  let resultado = a * b`<br>`  return resultado;`<br>`}`<br><br>`let resultado = multiplicar(4, 5);  //`<br>`Pasamos los valores 4 y 5 como argumentos`<br>`console.log(resultado);  // Output: 20` |
| **Función sin parámetros con retorno:** Esta es una función la cual no recibe ningún tipo de parámetro pero si genera algún retorno el cual puede ser guardado en alguna variable. | `function saludo() {`<br>`  return "Saludo desde funcion"`<br>`}`<br><br><br>`//Invocación de la función`<br>`let result = saludo()` |
| **Función flecha**: una función flecha (o arrow function) es una sintaxis más concisa y moderna para escribir funciones. Fue introducida en ECMAScript 6 (ES6) y se caracteriza por ser más breve que las funciones tradicionales, eliminando la necesidad de la palabra clave `function`, y permite escribir funciones más compactas. | `const saludar = () => "Somos Vankversity";`<br><br>`saludar()` |
| **Función flecha con parámetros** :Función flecha la cual se envían datos (parámetros) la cuales pueden ser usados en él bloque de código ejecutado. | `const sumar = (a, b) => a+b;`<br><br>`sumar(5, 3);  // Salida: 8` |
| **Función flecha con más de una instrucción:** Una función flecha con más de una instrucción en su cuerpo, debe implementar llaves({ }) y si hay un retorno, éste debe ponerse de forma explícita. | `const producto = (a, b) => {`<br>`  let resultado = a * b;`<br>`  let mensaje = `El resultado es`<br>`${resultado}`;`<br>`  return mensaje;`<br>`};`<br><br>`let resultado = producto(3, 5);`<br>`console.log(resultado);// Salida: El`<br>`resultado es 15` |

### Creación e Invocación:

Para crear una función usamos la palabra reservada function seguida del nombre de la función y los paréntesis, en cuales irían los parámetros si los hubiese.

### Sintaxis:

```
function nombreDeLaFuncion() {

    // Bloque de código a ejecutar

}
```

Para invocar nuestra función creada debemos escribir él nombre de nuestra función acompañado de paréntesis los cuales indican que está es una función.

### Ejemplo

```
function saludo() {

    console.log("Somos Vankversity");

    console.log("Somo el futuro");

}

saludo() //invocación de la función para que sea ejecutada
```

En esté ejemplo invocamos a la función saludo() para que se ejecute él bloque de código que tiene en su interior.

### Función con parámetros:

Una función con parámetros nos permite usar valores de entrada para la función con los cuales se podrán realizar acciones las cuales dependiendo de nuestras necesidades serán de utilidad. Los parámetros deben de estar indicados dentro de los paréntesis e ir separados por comas (,).

### Sintaxis:

```
function nombreDeLaFuncion(parametro1, parametro2) {

    // Bloque de código a ejecutar

}
```

### Ejemplo:

```
function dividir(dividendo, divisor) {

    console.log(`El resultado de la división es: ${dividendo/divisor}`);

}

dividir(10,2) //Salida: El resultado de la división es: 5
```

En este ejemplo, creamos la función dividir y como parámetros debemos enviar dos números los cuales serán él dividendo y él divisor respectivamente.

### Función con parámetros y retorno:

Está función además de contener parámetros de entrada también cuenta con un retorno, él cual es una respuesta que devolverá la función después de ejecutar las instrucciones dentro de ella, dicha respuesta la podemos guardar en una variable como se muestra a continuación.

### Sintaxis:

```
function nombreFuncion(parametro1, parametro2) {

    // Operaciones con el parámetro

    return resultado; // Respuesta que se envia despues de realizar el bloque de código

}
```

### Ejemplo:

```
function dividir(dividendo, divisor) {

    console.log(`El resultado de la división es: ${dividendo/divisor}`);

}

let resultado = dividir(10,2)

console.log(resultado); //Salida: El resultado de la división es: 5
```

### Problema de ejemplo: Determine el mayor de dos números usando funciones

Escriba una función que recibe por parámetros  dos numeros y retorne él número mayor, en caso de que los numeros sean iguales él programa debe retornar un -1.

**Solución:**

```
function determinarNumMayor(num1, num2){

    if(num1 === num2){

        return "Iguales"

    }else if(num1 > num2){

        return num1

    }else{

        return num2

    }

}// Fin de la función

let num1 = 15

let num2 = 34



let numMayor = determinarNumMayor(num1, num2)// Invocamos la función y el retorno lo guardamos
en numMayor

if(numMayor == "Iguales"){

    console.log("Ambos numeros son iguales");

}else{

    console.log(`El número mayor entre ${num1} y ${num2} es: ${numMayor}`);

}
```

### Función flecha:

Una función flecha es una forma más concisa, usa la sintaxis `=>`.

Esta al igual que una función normal puede o no contener parámetros de entrada y también retornar o no un valor.

### Sintaxis:

```
const nombreFuncion = (parámetro) =>  //código a ejecutar;
```

### Ejemplo:

```
const concatenador = (cadena1, cadena2) => cadena1 + " " + cadena2;

let cadenaConcatenada1 = concatenador("Somos", "Vankversity")

console.log(cadenaConcatenada1); // Salida: "Somos Vankversity"



let cadenaConcatenada2 = concatenador("Somos", "Programadores")

console.log(cadenaConcatenada2); // Salida: "Somos Programadores"
```

En este ejemplo, creamos una función llamada concatenador la cual recibe dos cadenas de entrada,la función se encarga de unir dichas cadenas y nos retorna el resultado de la unión.

**Problema de ejemplo: Calculadora de notas**

Escriba una función la cual reciba 5 notas de un estudiante y genere un mensaje informando si el estudiante aprobó o no la materia, para que la materia se de como aprobada el promedio del estudiante debe ser mayor o igual a 3,0.

**Resultado esperado:**

Mensaje afirmando o negando el aprobado de la materia

**Recursos a nuestra disposición:**

- nota1
- nota2
- nota3
- nota4
- nota5

**¿Cómo llegar al resultado esperado?**

Para llegar al resultado esperado el problema nos plantea que debemos generar el promedio de las 5 notas del estudiante, Entonces debemos comprender cómo realizar el promedio de las notas.

Para calcular el promedio de un conjunto de números, se suman todos los valores y luego se divide el resultado entre la cantidad total de elementos en el conjunto.

$$\bar{x} = \frac{a_1 + a_2 + a_3 + \ldots\ldots + a_n}{n}$$

Teniendo esto en cuenta ahora podemos empezar a realizar nuestro código.

Primero vamos a realizar nuestra función con los datos obligatorios para esta y en su interior realizaremos el cálculo comprobando si el promedio generado es mayor o igual a 3,0.
En dado caso que el promedio sea 3,0 o más generamos un mensaje indicando al usuario que el estudiante ha aprobado la materia, en el caso contrario generamos un mensaje indicando al usuario que el estudiante no ha aprobado la materia.
Nuestro código se vería algo así:

```
function calculadorNotas(nota1, nota2, nota3, nota4, nota5){
  let promedio = (nota1 + nota2 + nota3 + nota4 + nota5)/5

  if(promedio >= 3){

    console.log("El estudiante ha aprobado la materia");

  }else{

    console.log("El estudiante no ha aprobado la materia");

  }
}
```

Ahora debemos realizar la petición de las variables necesarias para la función al usuario, en este caso serían las notas del estudiante, La solicitud de estas notas se vería así:

```
let nota1 = parseFloat(prompt("Ingrese la nota 1"))
let nota2 = parseFloat(prompt("Ingrese la nota 2"))
let nota3 = parseFloat(prompt("Ingrese la nota 3"))
let nota4 = parseFloat(prompt("Ingrese la nota 4"))
let nota5 = parseFloat(prompt("Ingrese la nota 5"))
```

El código completo se vería así:

```
let nota1 = parseFloat(prompt("Ingrese la nota 1"));
let nota2 = parseFloat(prompt("Ingrese la nota 2"));
let nota3 = parseFloat(prompt("Ingrese la nota 3"));
let nota4 = parseFloat(prompt("Ingrese la nota 4”));
let nota5 = parseFloat(prompt("Ingrese la nota 5"));


function calculadorNotas(nota1, nota2, nota3, nota4, note 5){
  let promedio = (nota1 + nota2 + nota3 + nota4 + nota5)/5

  if(promedio >= 3){

    console.log(`El estudiante ha aprobado la materia su promedio fue: ${promedio}`);

  }else{

    console.log(`El estudiante no ha aprobado la materia su promedio fue: ${promedio}`);

  }
}

calculadorNotas(nota1, nota2, nota3, nota4, nota5);
```

**Problema de ejemplo: Calcular área y perímetro de un rectángulo.**
Escriba un programa que calcule el área y el perímetro de un rectángulo. Usa dos funciones:

calcularArea(altura, base): calcula el área del rectángulo.

calcularPerimetro(altura, base): calcula el perímetro.

**Resultado esperado:**

El area y perimetro de nuestro rectángulo ambos generados por funciones independientes.

**Recursos a nuestra disposición:**

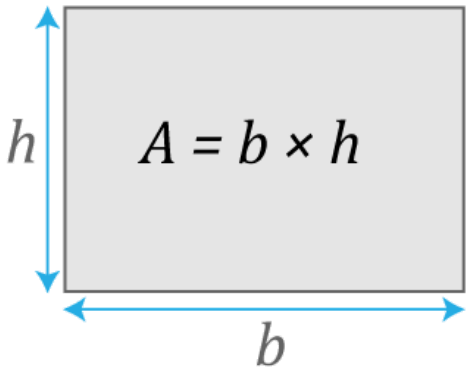
- Altura del rectángulo.
- Ancho del rectángulo.

**¿Cómo llegar al resultado esperado?**
Para llegar a nuestro resultado esperado primero debemos tener en cuenta 2 conceptos los cuales serán muy importantes a la hora de realizar nuestras funciones, dichos conceptos son como saber el área de un rectángulo y como saber el perímetro de un rectángulo.

**Área de un rectángulo**

El área de un rectángulo es la medida de la superficie que ocupa. Se calcula multiplicando la longitud de su base  por su altura.

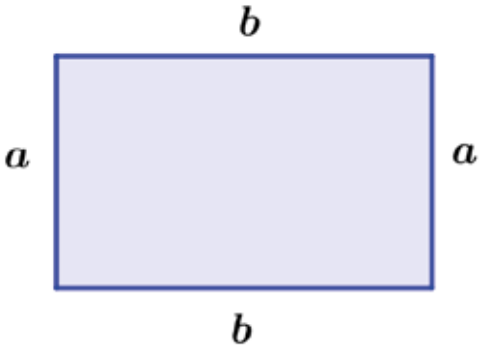
**Fórmula del área:**



### Perímetro de un rectángulo

El perímetro de un rectángulo es la medida total de la longitud de todos sus lados. Se calcula sumando la longitud de todos los lados, o multiplicando por dos la suma de la base y la altura.

**Fórmula del perímetro:**




$$p = 2(a + b)$$

Teniendo estos conceptos ahora más claros podemos empezar a realizar nuestro código.

Primero realizaremos nuestras funciones solicitadas en la problemática, una para calcular el área y otra para calcular el perímetro, Ambas recibirán por parámetros la altura y la base de nuestro rectángulo y nos retornaran un valor.

```
function calcularArea(altura, base){
  let area = base*altura
  return area
}



function calcularPërimetro(altura, base){
  let perimetro = 2*(base+altura)
  return perimetro
}
```

Ahora realicemos el bloque de código en el cual realizamos la petición de la altura y la base del rectangulo, esta se veria asi

```
let altura = parseFloat(prompt("Ingrese la altura de su rectangulo"))
let base = parseFloat(prompt("Ingrese la base de su rectangulo"))
```

Finalmente realizaremos las llamadas a las funciones enviando como argumentos la base y la altura que solicitamos al usuario, el código completo se vería así:

```
let altura = parseFloat(prompt("Ingrese la altura de su rectángulo"))
let base = parseFloat(prompt("Ingrese la base de su rectángulo"))

function calcularArea(altura, base){
  let area = base*altura
  return area
}

function calcularPerimetro(altura, base){
  let perimetro = 2*(base+altura)
  return perimetro
}

let area = calcularArea(altura, base)
let perimetro = calcularPerimetro(altura, base)
console.log(`El area de el rectangulo es: ${area} y su perimetro es: ${perimetro}`);
```