

# Técnicas de Programación

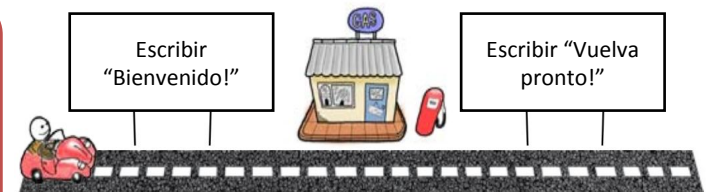
## CFP Programador full-stack

*Repaso General para el Examen*

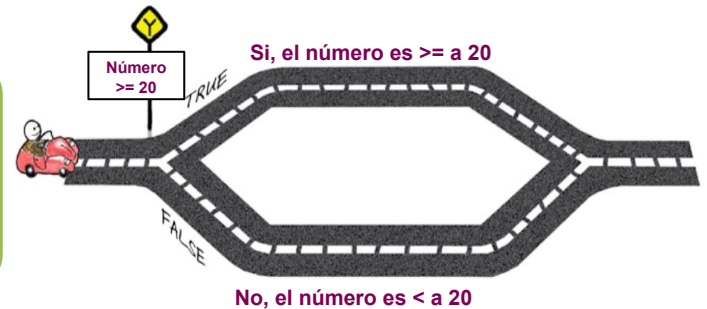
# Estructuras de Control

## Selección

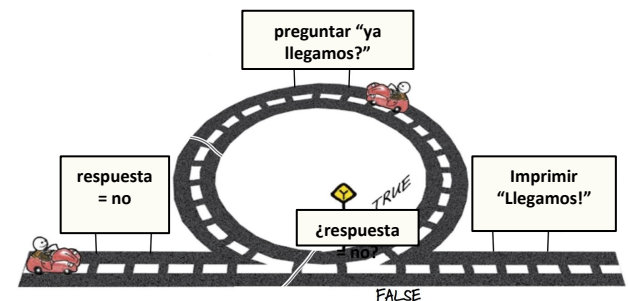
Secuenciales



Selección o de Decisión

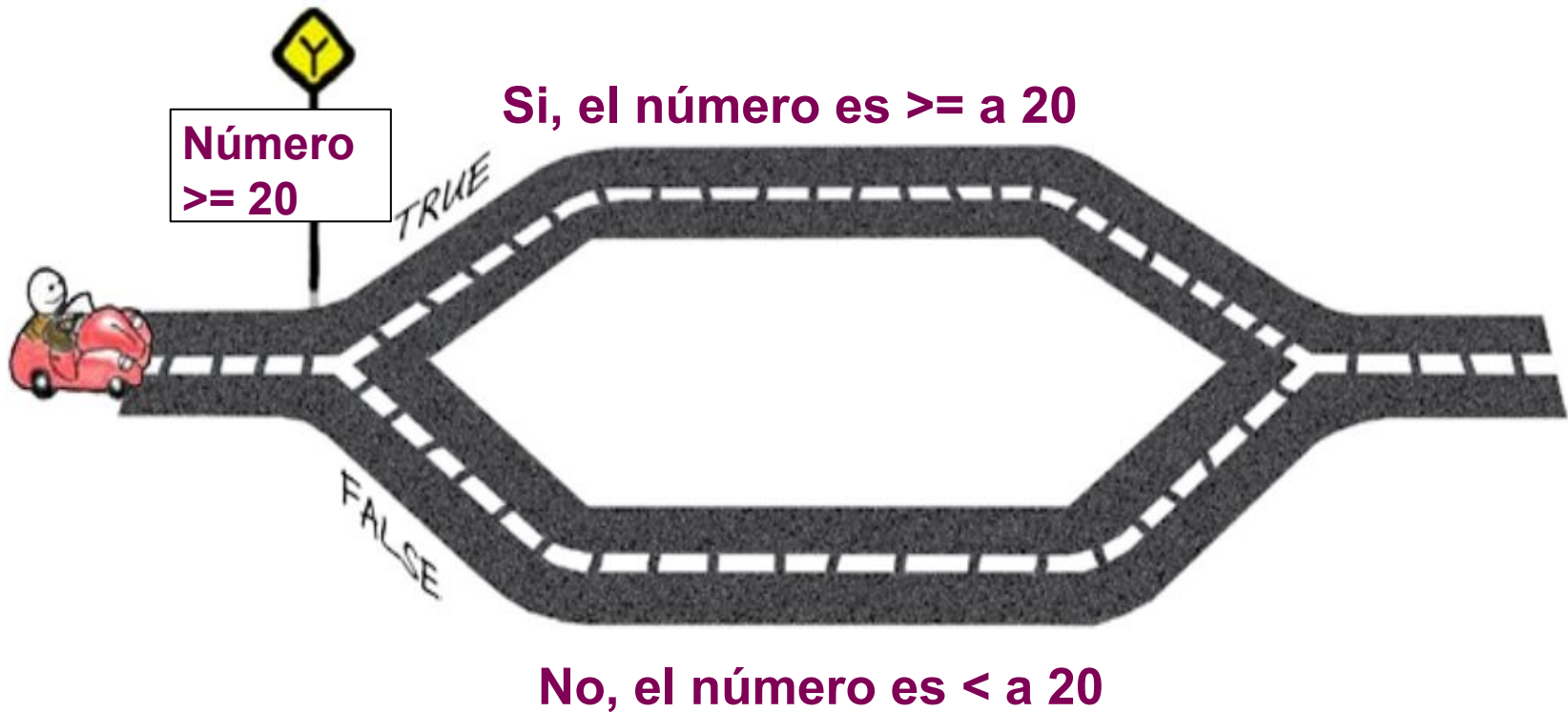


Repetitivas



# Estructuras de Control

## *Selección*

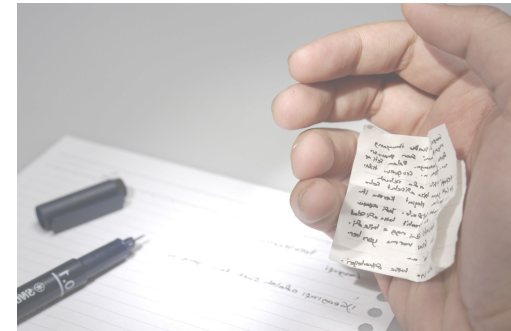


# Estructura de Control

## *Selección Simple y Múltiple*

### Selección Simple (Si)

```
if (<condición>) {  
    <instrucciones>  
} else {  
    <instrucciones>  
}
```

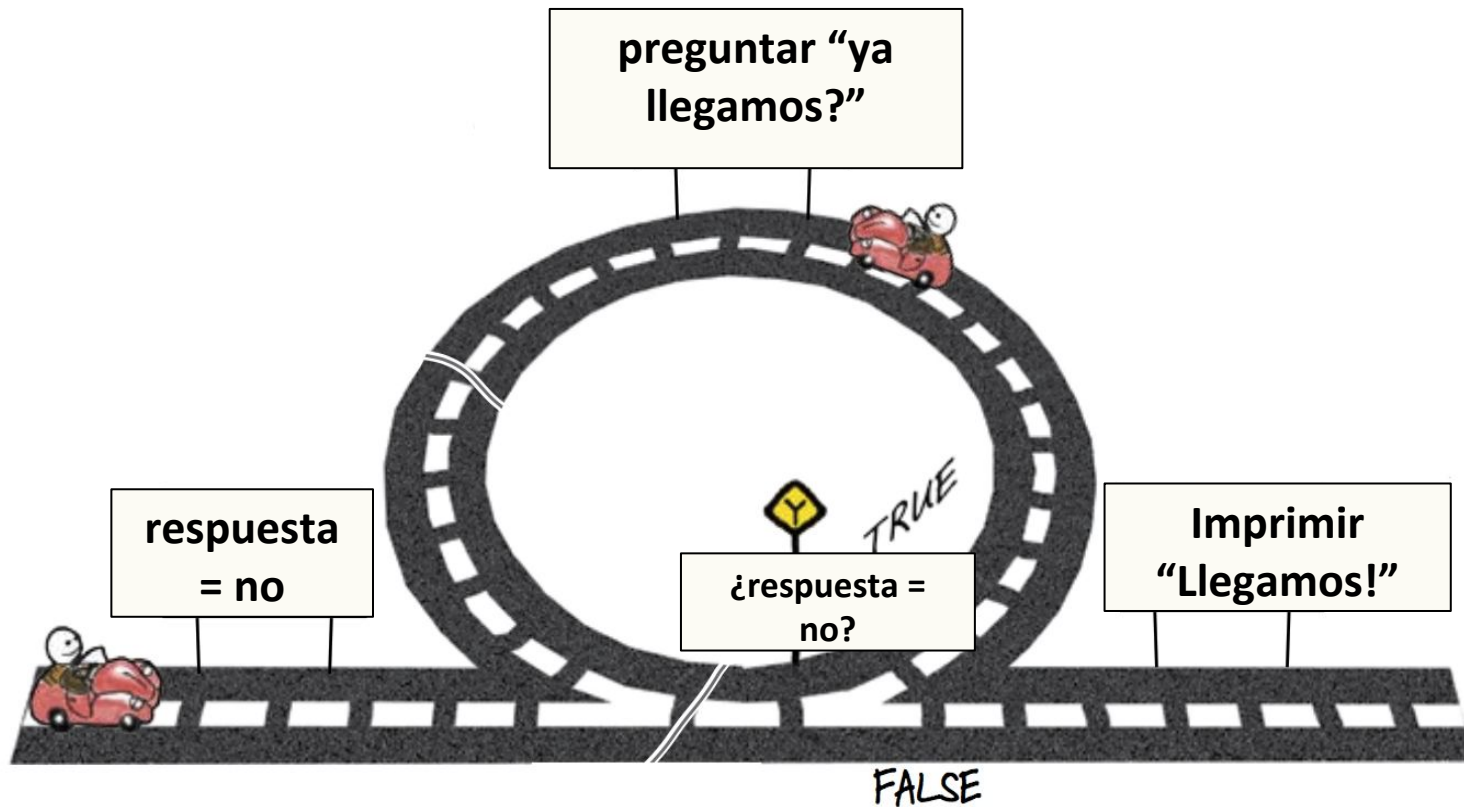


### Alternativa Múltiple (Según)

```
switch (<condición>) {  
    case <opción1>: <instrucciones> break;  
    case <opción2>: <instrucciones> break;  
    <...>  
    default: <instrucciones>  
}
```

# Estructuras de Control

## *Iteración / Repetición*



# Estructuras de Control

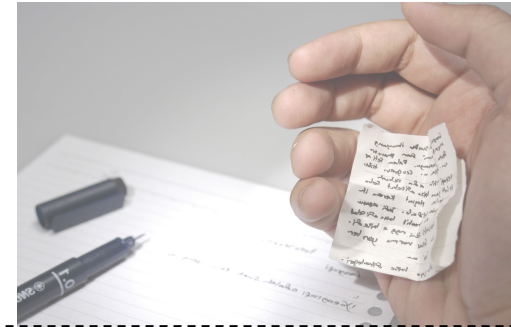
## *Iteración / Repetición*

Mientras  
(while)

```
while (<condición>) {  
    <instrucciones>  
}
```

Repetir  
(do while)

```
do {  
    <instrucciones>  
}  
while (<condición>)
```



Para  
(for)

```
for (<varctrl>=<inicial>; <varctrl> < <final>; <varctrl><in/decr> ) {  
    <instrucciones>  
}
```

# Operadores Condicionales

Operador	Significado	Ejemplo
>	Mayor que	$3 > 1$
<	Menor que	$1 < 3$
==	Igual que	$1 == 1$
>=	Mayor igual que	$4 >= 2$
<=	Menor igual que	$4 <= 2$
<>	Distinto que	$9 <> 3$

# Operadores Lógicos

Operador	Significado	Descripción	Ejemplo
&&	Conjunción (Y)	Ambas son Verdaderas	$(7 > 4) \ \&\& \ (2 == 2)$
	Disyunción (O)	Al menos una es verdadera	$(1 == 1 \    \ 2 == 1)$
!	Negación (No)	No es verdadero	$!(2 < 5)$



# Prueba de Escritorio

- Técnica utilizada para validar la resolución de problemas con algoritmos, de uso frecuente en el ámbito informático
- Sirve para validar utilizando datos reales como ejemplo, un algoritmo definido y así comprobar si se obtiene el resultado deseado
- Ejemplo, recuerde el ejercicio de verificar si un número es mayor a 20. Se podría verificar con un número mayor a 20, un número igual a 20 y un número menor que 20

# Estructura de Control - Selección

*Mayor a 20 - Prueba de Escritorio*

Código	Datos Entrada	Respuesta Deseada
<pre>//Algoritmo Mayor20  let readlineSync = require('readline-sync');  let nroDeseado; nroDeseado=readlineSync.questionInt("Escriba el número que desea verificar si es mayor o no a 20: ");  if (nroDeseado &gt; 20) {   console.log('El número es mayor a 20: ',nroDeseado); } else {   console.log('El número es menor o igual a 20: ',nroDeseado); }</pre>	nroDeseado = 20	El número es menor o igual a 20: 20
	nroDeseado = 3	El número es menor o igual a 20: 3
	nroDeseado = 45	El número es mayor a 20: 45

# Métodos


- **Agrupar** un conjunto de sentencias de código **cohesivas**
- Tienen un **nombre representativo**
- Pueden ser invocados
- Pueden declarar parámetros
- Pueden devolver un valor
- Nos ayudan a **reusar** el código



# Métodos

- Cada vez que se encuentra una llamada a un **método**:
  - El programa ejecuta el código del método hasta que termina
  - Vuelve a la siguiente línea del lugar donde partió

```
if (opcionMenu==1) {  
  dibujarGuiones  
  console.log("El resultado de la  
operacion es: ", numero1+numero2);  
}  
  
function (dibujarGuiones) {  
  let x;  
  for (x=1; x<=40; x++) {  
    console.log("-");  
  }  
  console.log("\n");  
}
```



# Métodos con Retorno

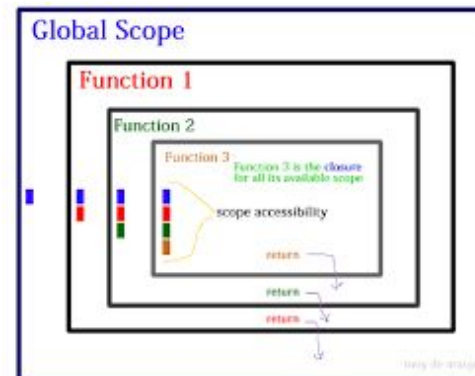
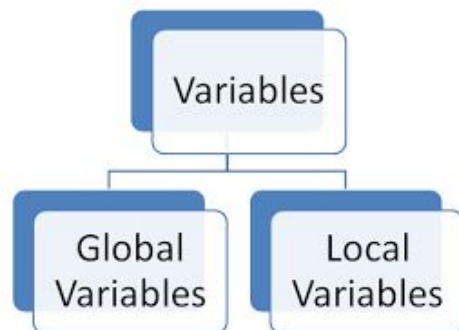
- Análogamente se puede utilizar para retornar algun valor

```
function nombre_del_metodo(argumento_1,argumento_2,... ) {  
    let retorno;  
    acción 1;  
    acción ...;  
    acción n;  
    return retorno;  
}
```

# Ámbito de las Variables

Al utilizar funciones se establece un límite para el alcance de las variables

- **Variables Locales:** Son aquellas que se encuentran dentro de un método. El valor se confina al método en el que está declarada
- **Variables Globales:** Son las que se definen o están declaradas en el algoritmo principal. Pueden utilizarse en cualquier método
- Se debe intentar crear métodos con variables locales y pocos parámetros para favorecer la reutilización y el mantenimiento del software



# Buenas Prácticas de Programación

## Entender el problema, diseñar una estrategia, implementar

- Nombres representativos de variables y métodos
- Código claro, comprensible, etc.
- Identación en las estructuras de control
- Comentarios en el código
- *//Así se comenta en javascript, con las dos barras*



# Buenas Prácticas de Programación

- Usar métodos
- No duplicar código
- Dividir el problema en sub problemas
- Construir el código tan simple como sea posible
- Que el código funcione no significa que esté bien programado



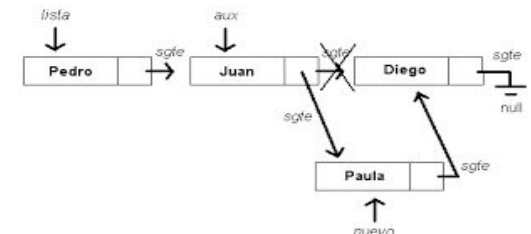
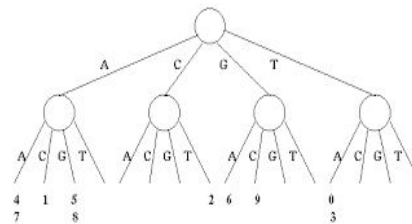


# Estructuras de Datos

## Forma particular de organizar datos



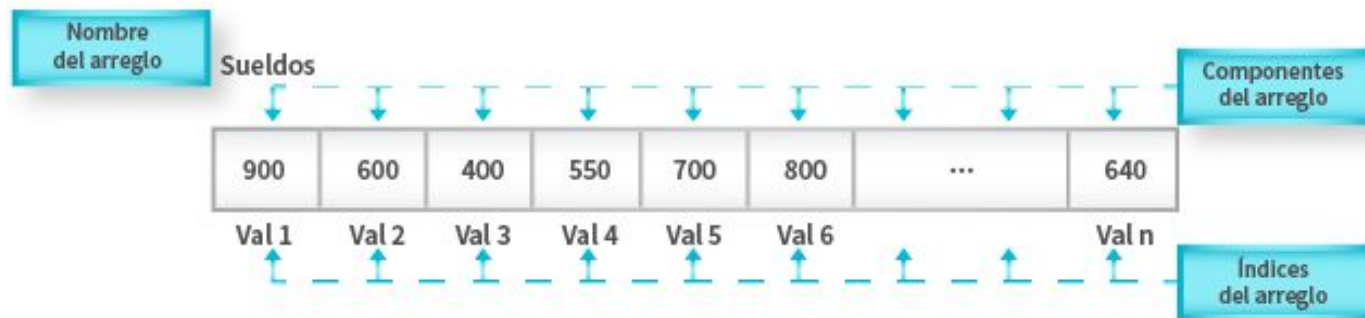
- Estructuras que permiten **COLECCIONAR** elementos
    - GUARDARLOS
    - RECORRERLOS
    - MANIPULARLOS
  - Operaciones básicas
    - **COLOCAR**
    - **OBTENER**
- Diagrama de un árbol de búsqueda binario. El nodo raíz tiene cuatro hijos etiquetados A, C, G y T. Cada uno de estos nodos hijos tiene a su vez cuatro hijos más, etiquetados A, C, G y T. Debajo de cada uno de estos nodos de nivel 2 se encuentran números: el primer grupo (A, C, G, T) tiene 4, 1, 5, 8; el segundo (A, C, G, T) tiene 2, 6, 9; el tercero (A, C, G, T) tiene 0, 3.
- Estructuras
    - **LISTAS**
    - **COLAS**
    - **PILAS**
    - **ARBOLES**
- Diagrama de una lista enlazada. Se muestran dos nodos rectangulares. El primer nodo contiene "Pedro" y tiene una flecha que apunta al segundo nodo. El segundo nodo contiene "Juan" y tiene una flecha que apunta a un tercer nodo (parcialmente visible). Encima del primer nodo hay una etiqueta "lista" con una flecha que apunta al nodo. Encima del segundo nodo hay una etiqueta "aux" con una flecha que apunta al nodo. Entre los dos nodos hay una etiqueta "sigte" con una flecha que apunta a la flecha del primer nodo.



# Estructuras de Datos

## *Arreglos / Listas / Vectores*

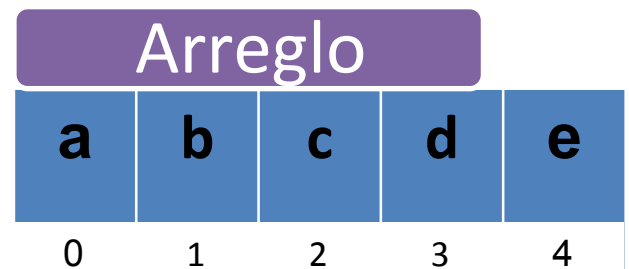
- Los arreglos son estructuras de datos homogéneas (todos sus datos son del mismo tipo)
- Permiten almacenar un determinado número de datos
- Tiene muchos elementos, y a cada uno de ellos se acceden indicando que posición se quiere usar (un índice)



# Estructuras de Datos

## *Arreglos / Listas / Vectores*

- Lista = Array
- Los elementos deben ser del mismo tipo de dato
- Zero-based (arreglos de base cero) -> Índices comienzan en 0
- La cantidad de elementos total = Length será igual al número del último elemento más 1
- Propiedades:
  - ELEMENTO o ITEM: a, b, c, d, e
  - LONGITUD: 5
  - INDICE o SUBINDICE: 0, 1, 2, 3, 4



**Longitud = Length = 5**

# Estructuras de Datos – Arreglos

## *Ejercicio – Arreglo de Números*

- Crear un arreglo llamado num que almacene los siguientes datos: 20, 14, 8, 0, 5, 19 y 24 y se los muestre al usuario
- Al utilizar arreglos en base cero los elementos validos van de 0 a  $n-1$ , donde  $n$  es el tamaño del arreglo
- En el ejemplo 1 las posiciones / índice del num entonces van desde 0 a  $7-1$ , es decir de 0 a 6

	num						
Datos del arreglo	20	14	8	0	5	19	24
Posiciones	0	1	2	3	4	5	6

# Estructuras de Datos – Arreglos

## *Ejercicio – Arreglo de Números - Código*

//Algoritmo ArregloNumeros

```
let num = new Array(7);
```

```
let indice;
```

```
num[1] = 14;  
num[2] = 8;  
num[3] = 0;  
num[4] = 5;  
num[5] = 19;  
num[6] = 4;
```

```
indice = 0;
```

```
while (indice < 7) {  
  console.log("El número en la posición ", indice, " es ", num[indice]);  
  indice++;  
}
```

Definición del arreglo num  
con dimensión 7

# Estructuras de Datos – Arreglos

## *Ejercicio – Arreglo de Números - Código*

//Algoritmo ArregloNumeros

```
let num = new Array(7);
```

```
let indice;
```

```
num[1] = 14;  
num[2] = 8;  
num[3] = 0;  
num[4] = 5;  
num[5] = 19;  
num[6] = 4;
```

```
indice = 0;
```

```
while (indice < 7) {  
  console.log("El número en la posición ", indice, " es ", num[indice]);  
  indice++;  
}
```

Se completa el arreglo con  
números fijos

# Estructuras de Datos – Arreglos

## Ejercicio – Arreglo de Números - Código

//Algoritmo ArregloNumeros

```
let num = new Array(7);
```

```
let indice;
```

```
num[1] = 14;  
num[2] = 8;  
num[3] = 0;  
num[4] = 5;  
num[5] = 19;  
num[6] = 4;
```

```
indice = 0;
```

```
while (indice < 7) {  
  console.log("El número en la posición ", indice, " es ", num[indice]);  
  indice++;  
}
```

Se inicializa el índice para comenzar a recorrer el arreglo desde la posición 0



# Estructuras de Datos – Arreglos

## Ejercicio – Arreglo de Números - Código

//Algoritmo ArregloNumeros

```
let num = new Array(7);
```

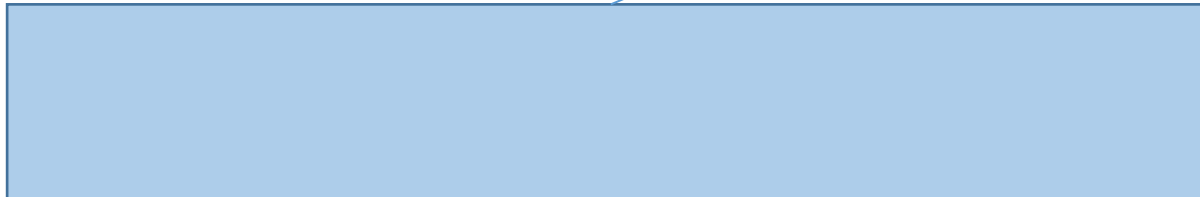
```
let indice;
```

```
num[1] = 14;  
num[2] = 8;  
num[3] = 0;  
num[4] = 5;  
num[5] = 19;  
num[6] = 4;
```

```
indice = 0;
```

```
while (indice < 7) {  
  console.log("El número en la posición ", indice, " es ", num[indice]);  
  indice++;  
}
```

Recorre el arreglo mostrando  
los números que posee





# Estructuras de Datos – Arreglos

## *Ejercicio – Arreglo de Números - Código*

//Algoritmo ArregloNumeros

```
let num = new Array(7);
```

```
let indice;
```

```
num[1] = 14;  
num[2] = 8;  
num[3] = 0;  
num[4] = 5;  
num[5] = 19;  
num[6] = 4;
```

```
indice = 0;
```

```
while (indice < 7) {  
  console.log("El número en la posición ", indice, " es ", num[indice]);  
  indice++;  
}
```

# Estructuras de Datos – Arreglos

## Ejercicio – Arreglo de Números - Código

//Algoritmo ArregloNumeros

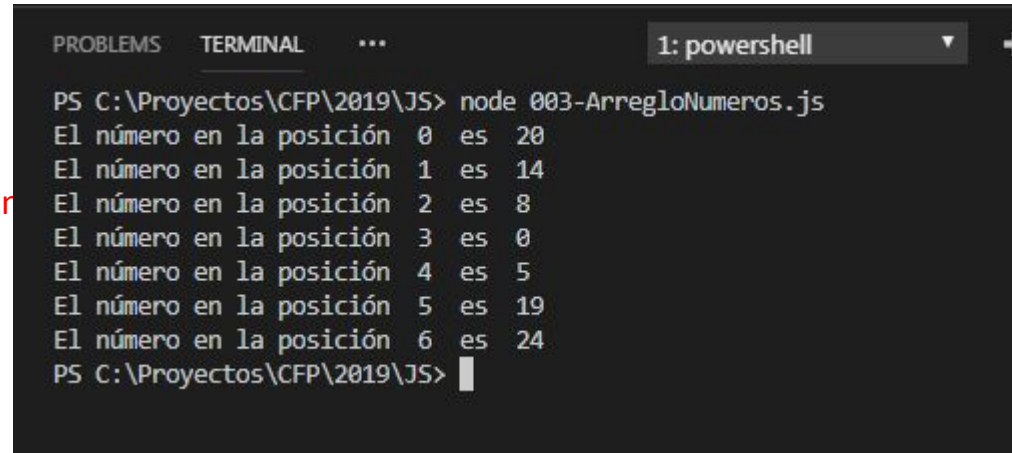
```
let num = new Array(7);
```

```
let indice;
```

```
num[1] = 14;  
num[2] = 8;  
num[3] = 0;  
num[4] = 5;  
num[5] = 19;  
num[6] = 4;
```

```
indice = 0;
```

```
while (indice < 7) {  
  console.log("El número en la posición  
  indice++;  
}
```



```
PROBLEMS  TERMINAL  ...  1: powershell  
PS C:\Proyectos\CFP\2019\JS> node 003-ArregloNumeros.js  
El número en la posición 0 es 20  
El número en la posición 1 es 14  
El número en la posición 2 es 8  
El número en la posición 3 es 0  
El número en la posición 4 es 5  
El número en la posición 5 es 19  
El número en la posición 6 es 24  
PS C:\Proyectos\CFP\2019\JS>
```

# Matrices

- Permiten representar más de 1 dimensión (a diferencia de los arreglos)
- Si tienen 2 dimensiones, son como tablas (n filas y m columnas)
- Si tienen 3 dimensiones, son como espacios con ancho, alto y profundidad (X, Y, Z)
- En javascript no existen los arreglos multidimensionales, estos se definen anidando arreglos dentro de arreglos.

$$A = \begin{bmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{i1} & b_{i2} & b_{i3} & \dots & b_{in} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & b_{m3} & \dots & b_{mn} \end{bmatrix}$$

n Columnas

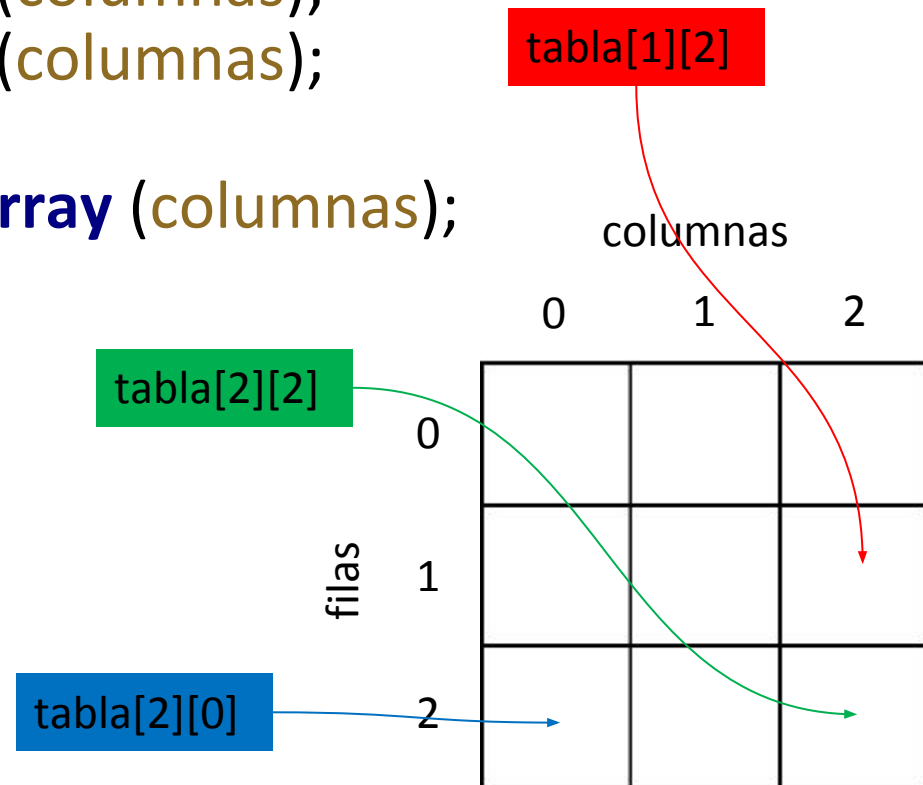
m Filas

# Matrices

```
identificador = new Array (filas);  
identificador[0] = new Array (columnas);  
identificador[1] = new Array (columnas);  
...  
identificador[filas-1] = new Array (columnas);
```

• *Ejemplo:* tabla (3,3)

```
tabla = new Array (3);  
tabla[0] = new Array (3);  
tabla[1] = new Array (3);  
tabla[2] = new Array (3);
```



# Matrices

**Para recorrer una matriz necesitamos 2 índices**

```
let fila, columna;
for (fila = 0 ; fila < nroFilas ; fila++) {
    for (columna = 0 ; columna < nroColumnas ; columna++) {
        console.log (matriz[fila][columna], " ");
    }
}
```

# Estructuras de Datos

## *Arreglos, Métodos y Pasaje de Parámetros*

- Podemos **reutilizar** código!
- Las **modificaciones** se pueden hacer **directamente** en los arreglos que pasamos como **parámetro** (solo funciona para arreglos y matrices, no para otros tipos de datos)



# Estructuras de Datos y Métodos

*Definición con Estructuras como Parámetros*

```
function contarCeros (v, cantidad) {  
  let contador = 0;  
  let indice;  
  for (indice = 0; indice < cantidad; indice++) {  
    if (v[indice] == 0) {  
      contador++;  
    }  
  }  
  return contador;  
}
```

Al pasar como parámetro un arreglo, también debo indicar su dimensión para poder recorrer todos sus valores

# Estructuras de Datos y Métodos

## *Retornos de Arreglos/Matrices*

```
let readlineSync = require('readline-sync');
```

```
function cargarVector(v, cantidad)
```

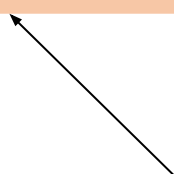
```
    let indice;
```

```
    for (indice = 0; indice < cantidad; indice++) {
```

```
        v[indice]=readlineSync.questionInt("Ingrese el valor ", indice, ":");
```

```
    }
```

```
}
```



Las modificaciones se hacen sobre el arreglo declarado como parámetro que es el arreglo original



# Estructuras de Datos

## *Pasos para Migrar a Métodos*

1. Identificar código repetido o funcionalidad “reusable”
2. Identificar parámetros comunes y retorno (si fuese necesario devolver un resultado)
3. Modificar el código para aprovechar el código mejorado (por ejemplo, la carga de un vector o la escritura por pantalla)



# Ejercicios de Repaso

## *Ejercicio – Calcular Promedio*

- El DT de los infantiles del equipo de futbol desea saber el promedio de la edad de los chicos
- La edad de los chicos va de 3 a 7 años. Las edades es cargada al azar (use la función `aleatorio(menorValor,mayorValor)`, es decir `aleatorio(3,7)`)
- Muestre todas las edades y el promedio de las mismas


$$\begin{array}{l} 6 + 8 + 9 + 6 + 9 + 7 + 8 + 5 = 58 \\ \text{Icon 1} + \text{Icon 2} + \text{Icon 3} + \text{Icon 4} + \text{Icon 5} + \text{Icon 6} + \text{Icon 7} + \text{Icon 8} = 8 \\ \hline 58 \div 8 = 7.2 \end{array}$$

# Ejercicios de Repaso

## *Ejercicio – Calcular Promedio*

//Genera un numero aleatorio entre min y max

```
function Aleatorio(min, max) {  
    return Math.floor(Math.random() * (max - min) ) + min;  
}
```

//Calcula el promedio de las edades de los jugadores de futbol

```
let readlineSync = require('readline-sync');
```

```
let promedio = 0;
```

```
let dimArreglo = readlineSync.questionInt("Indique la cantidad de jugadores: " );
```

```
let numArreglo = new Array (dimArreglo);
```

```
cargarArreglo(numArreglo, dimArreglo);
```

```
mostrarArreglo(numArreglo, dimArreglo);
```

```
promedio = obtenerPromedio(numArreglo, dimArreglo);
```

```
console.log ("El promedio las edades es de: ", promedio);
```


$$\begin{array}{l} 6 + 8 + 9 + 6 + 9 + 7 + 8 + 5 = 58 \\ \text{8 jugadores} = \frac{58}{8} = 7.2 \end{array}$$

# Ejercicios de Repaso

## *Ejercicio – Calcular Promedio - Errores*



The diagram illustrates the calculation of an average. The top row shows a sequence of numbers in colored boxes: 6 (blue), 8 (orange), 9 (green), 6 (purple), 9 (green), 7 (pink), 8 (blue), and 5 (pink). These are summed to get 58. The bottom row shows eight stick figures of different colors, representing the count of 8. The final result is 7.2, shown in red.

$$6 + 8 + 9 + 6 + 9 + 7 + 8 + 5 = 58$$

$$\frac{58}{8} = 7.2$$



//Completa un arreglo con números aleatorios del 3 al 7

```
function cargarArreglo(numeroArreglo,dimensionArreglo) {
    for ( indice = 0 ; indice < dimensionArreglo; indice++) {
        numeroArreglo[indice] = Aleatorio(3,7);
    }
}
```

# Ejercicios de Repaso

## Ejercicio – Calcular Promedio - Errores

$$\begin{array}{ccccccccccccc} \boxed{6} & + & \boxed{8} & + & \boxed{9} & + & \boxed{6} & + & \boxed{9} & + & \boxed{7} & + & \boxed{8} & + & \boxed{5} & = & 58 \\ \text{boy} & & \text{girl} & & \text{boy} & & \text{girl} & & \text{boy} & & \text{girl} & & \text{boy} & & \text{girl} & & \\ \text{boy} & + & \text{girl} & + & \text{boy} & + & \text{girl} & + & \text{boy} & + & \text{girl} & + & \text{boy} & + & \text{girl} & = & \frac{58}{8} = 7.25 \end{array}$$



//Completa un arreglo con números aleatorios del 3 al 7

```
function cargarArreglo(numeroArreglo,dimensionArreglo) {  
    for ( indice = 0; indice < dimensionArreglo; indice++) {  
        numeroArreglo[indimensionArreglo] = Aleatorio(3,7);  
    }  
}
```

**Falta definir indice**

**No va  
dimensionArreglo  
va indice**

# Ejercicios de Repaso

## *Ejercicio – Calcular Promedio*



The diagram illustrates the calculation of an average. The top row shows a sequence of eight numbers in colored boxes: 6 (blue), 8 (orange), 9 (green), 6 (purple), 9 (green), 7 (pink), 8 (blue), and 5 (pink). These are summed to equal 58. The bottom row shows eight stick figures of different colors corresponding to the numbers above. The sum of these figures is divided by 8, resulting in an average of 7.2.

$$6 + 8 + 9 + 6 + 9 + 7 + 8 + 5 = 58$$
$$\frac{58}{8} = 7.2$$

//Completa un arreglo con números aleatorios del 3 al 7

```
function cargarArreglo(numeroArreglo,dimensionArreglo) {  
    let indice;  
    for ( indice = 0 ; indice < dimensionArreglo; indice++) {  
        numeroArreglo[indice ] = Aleatorio(3,7);  
    }  
}
```

# Ejercicios de Repaso

## *Ejercicio – Calcular Promedio - Errores*

$$\begin{array}{l} 6 + 8 + 9 + 6 + 9 + 7 + 8 + 5 = 58 \\ \text{8 figures} = 8 \\ \hline 7.2 \end{array}$$

//Muestra un arreglo

```
function mostrarArreglo(numeroArreglo, dimensionArreglo) {  
    let numeroArreglo;  
    for ( indice = 0 ; indice < dimensionArreglo ; indice++) {  
        console.log ( " ", numeroArreglo[indice] );  
    }  
    console.log ( "\n " );  
}
```



# Ejercicios de Repaso

## Ejercicio – Calcular Promedio - Errores

$$6 + 8 + 9 + 6 + 9 + 7 + 8 + 5 = 58$$
$$\text{8 figures} = 8$$
$$= 7.2$$

//Muestra un arreglo

```
function mostrarArreglo(numeroArreglo, dimensionArreglo) {  
    let numeroArreglo;  
    for (indice = 0; indice < dimensionArreglo ; indice++) {  
        console.log (" ", numeroArreglo[indice] );  
    }  
    console.log ("\n ");  
}
```

Los parametros  
no se definen

Falta definir  
indice





# Ejercicios de Repaso

## *Ejercicio – Calcular Promedio*

$$6 + 8 + 9 + 6 + 9 + 7 + 8 + 5 = 58$$
$$\text{Figure 1} + \text{Figure 2} + \text{Figure 3} + \text{Figure 4} + \text{Figure 5} + \text{Figure 6} + \text{Figure 7} + \text{Figure 8} = 8$$
$$\frac{58}{8} = 7.2$$

//Muestra un arreglo

```
function mostrarArreglo(numeroArreglo, dimensionArreglo) {  
    let indice;  
    for ( indice = 0 ; indice < dimensionArreglo ; indice++) {  
        console.log ( " ", numeroArreglo[indice] );  
    }  
    console.log ( "\n " );  
}
```

# Ejercicios de Repaso

## *Ejercicio – Calcular Promedio*

//Completa un arreglo con números aleatorios del 3 al 7

```
function cargarArreglo(numeroArreglo,dimensionArreglo) {
    let indice;
    for ( indice = 0 ; indice < dimensionArreglo; indice++) {
        numeroArreglo[indice ] = Aleatorio(3,7);
    }
}
```

$$6 + 8 + 9 + 6 + 9 + 7 + 8 + 5 = 58$$

$$\frac{58}{8} = 7.2$$

//Muestra un arreglo

```
function mostrarArreglo(numeroArreglo, dimensionArreglo) {
    let indice;
    for ( indice = 0 ; indice < dimensionArreglo ; indice++) {
        console.log ( " ", numeroArreglo[indice] );
    }
    console.log ("\n ");
}
```

# Ejercicios de Repaso

## *Ejercicio – Calcular Promedio - Errores*

// Calcula el promedio de un arreglo

```
function obtenerPromedio(numArreglo, dimArreglo) {  
    let prome = 0;  
    let sumaTotal = 0;  
    let indice;  
    for (indice=0; indice < dimArreglo; indice++) {  
        sumaTotal = sumaTotal+numArreglo[indice];  
    }  
    prome=sumaTotal/dimArreglo;  
    return prome;  
}
```




$$6 + 8 + 9 + 6 + 9 + 7 + 8 + 5 = 58$$
$$\frac{58}{8} = 7.25$$

# Ejercicios de Repaso

## *Ejercicio – Calcular Promedio - Errores*

// Calcula el promedio de un arreglo

```
function obtenerPromedio(numArreglo, dimArreglo) {  
    let prome = 0;  
    let sumaTotal = 0;  
    let indice;  
    for (indice=0; indice < dimArreglo; indice++) {  
        sumaTotal = sumaTotal+numArreglo[dimArreglo];  
    }  
    prome=sumaTotal/dimArreglo;  
    return prome;  
}
```

No va  
dimArreglo  
va indice



The diagram illustrates the calculation of an average. The top row shows a sequence of numbers in colored boxes: 6 (blue), 8 (orange), 9 (green), 6 (purple), 9 (green), 7 (pink), 8 (blue), and 5 (pink). These are summed to get 58. The bottom row shows eight stick figures of different colors, representing the count of 8. The final result is 7.2, shown in red.

$$6 + 8 + 9 + 6 + 9 + 7 + 8 + 5 = 58$$
$$\frac{58}{8} = 7.2$$

# Ejercicios de Repaso

## *Ejercicio – Calcular Promedio*

// Calcula el promedio de un arreglo

```
function obtenerPromedio(numArreglo, dimArreglo) {  
    let prome = 0;  
    let sumaTotal = 0;  
    let indice;  
    for (indice=0; indice < dimArreglo; indice++) {  
        sumaTotal = sumaTotal+numArreglo[indice];  
    }  
    prome=sumaTotal/dimArreglo;  
    return prome;  
}
```

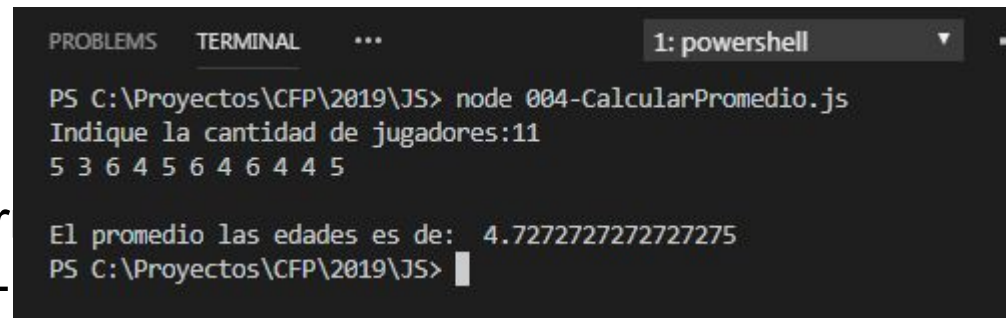

$$\begin{array}{l} 6 + 8 + 9 + 6 + 9 + 7 + 8 + 5 = 58 \\ \text{Figure 1} + \text{Figure 2} + \text{Figure 3} + \text{Figure 4} + \text{Figure 5} + \text{Figure 6} + \text{Figure 7} + \text{Figure 8} = 8 \\ \hline 58 \div 8 = 7.2 \end{array}$$

# Ejercicios de Repaso

## *Ejercicio – Calcular Promedio*

// Calcula el promedio de un arreglo

```
function obtenerPromedio(numArreglo, dimArreglo) {  
    let prome = 0;  
    let sumaTotal = 0;  
    let indice;  
    for (indice=0; indice < dimArreglo; indice++) {  
        sumaTotal = sumaTotal + numArreglo[indice];  
    }  
    prome = sumaTotal / dimArreglo;  
    return prome;  
}
```



The screenshot shows a PowerShell terminal window with the following content:

```
PROBLEMS  TERMINAL  ...  1: powershell  
PS C:\Proyectos\CFP\2019\JS> node 004-CalcularPromedio.js  
Indique la cantidad de jugadores:11  
5 3 6 4 5 6 4 6 4 4 5  
  
El promedio las edades es de: 4.7272727272727275  
PS C:\Proyectos\CFP\2019\JS>
```



The image shows a visual representation of the average calculation. The top row displays the sum of eight numbers: 6 + 8 + 9 + 6 + 9 + 7 + 8 + 5 = 58. The bottom row displays the sum of eight stick figures, each representing a number from the top row, with the result 8. The final result is 7.2, shown in red.

$$6 + 8 + 9 + 6 + 9 + 7 + 8 + 5 = 58$$
$$\frac{58}{8} = 7.2$$

# Ejercicios de Repaso

## *Ejercicio – Sumar Tres Arreglos*

- Hacer la suma de tres arreglos y dejarlo en otro arreglo
- La dimensión de los arreglos es solicitada al usuario
- Los dos arreglos son cargados al azar

# Ejercicios de Repaso

## *Ejercicio – Sumar Tres Arreglos*

```
// Calcula la suma de tres arreglos
let readlineSync = require('readline-sync');

let promedio = 0;
let dim = readlineSync.questionInt("Ingrese la dimensión del arreglo: ");
let arreglo1 = new Array (dim);
let arreglo2 = new Array (dim);
let arreglo3 = new Array (dim);
cargarArreglo(arreglo1, dim);
cargarArreglo(arreglo2, dim);
sumarArreglos(arreglo1, arreglo2, arreglo3, dim);
mostrarArreglo(arreglo1, dim);
mostrarArreglo(arreglo2, dim);
console.log ("La suma de los arreglos es:", mostrarArreglo(arreglo3, dim) );
```



# Ejercicios de Repaso

## *Ejercicio – Sumar Tres Arreglos - Errores*

//Completa un arreglo con números al azar menores que 100

```
function cargarArreglo(arreglo, dim) {  
    for ( indice = 0 ; indice < dim; indice++) {  
        arreglo[indice] = Math.random(100);  
    }  
}
```

// Suma dos arreglos y el resultado lo pone en otro arreglo

```
function sumarArreglos(arreglo1, arreglo2, arreglo3, dim) {  
    let indice;  
    for ( indice = 0 ; indice <= dim; indice++) {  
        arreglo3[indice] = arreglo1[indice] + arreglo2[indice] ;  
    }  
}
```



# Ejercicios de Repaso

## Ejercicio – Sumar Tres Arreglos - Errores

//Completa un arreglo con números al azar menores que 100

```
function cargarArreglo(arreglo, dim) {
  for ( indice = 0 ; indice < dim; indice++) {
    arreglo[dim] = Math.random(100);
  }
}
```

No define  
indice

No va dim  
va indice



Es <

No es  
arreglo3 es  
arreglo2

// Suma dos arreglos y el resultado lo pone en otro arreglo

```
function sumarArreglos(arreglo1, arreglo2, arreglo3, dim) {
  let indice;
  for ( indice = 0 ; indice <= dim; indice++) {
    arreglo3[indice] = arreglo1[indice] + arreglo2[indice] ;
  }
}
```

# Ejercicios de Repaso

## *Ejercicio – Sumar Tres Arreglos*

//Completa un arreglo con números aleatorios del 3 al 7

```
function cargarArreglo(arreglo, dim) {  
  let indice;  
  for ( indice = 0 ; indice < dim; indice++) {  
    arreglo[indice] = Math.random(100);  
  }  
}
```

// Suma dos arreglos y el resultado lo pone en otro arreglo

```
function sumarArreglos(arreglo1, arreglo2, arreglo3, dim) {  
  let indice;  
  for ( indice = 0 ; indice < dim; indice++) {  
    arreglo3[indice] = arreglo1[indice] + arreglo2[indice] ;  
  }  
}
```

# Ejercicios de Repaso

## *Ejercicio – Sumar Tres Arreglos*

//Muestra los elementos de un arreglo

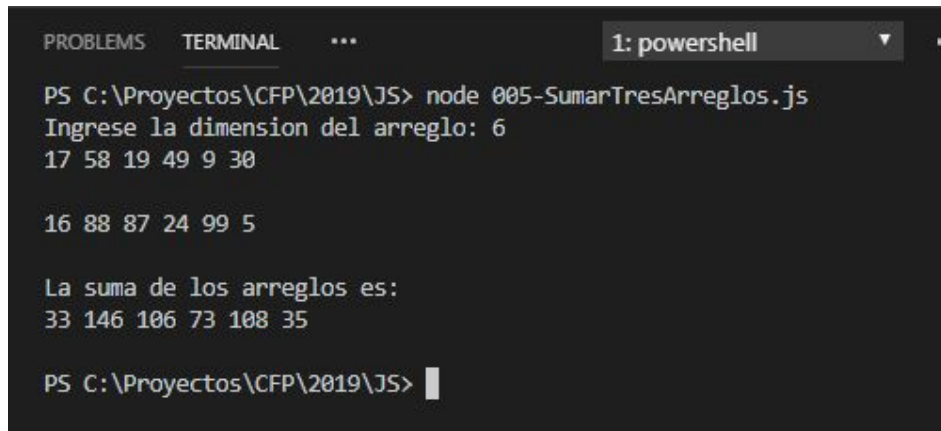
```
function mostrarArreglo(arreglo, dim) {  
    let indice;  
    let salida;  
    for ( indice = 0 ; indice < dim ; indice++) {  
        salida = salida + " " + numeroArreglo[indice];  
    }  
    console.log (salida);  
}
```

# Ejercicios de Repaso

## *Ejercicio – Sumar Tres Arreglos*

//Muestra los elementos de un arreglo

```
function mostrarArreglo(arreglo, dim) {  
    let indice;  
    let salida;  
    for ( indice = 0 ; indice < dim ; indice++) {  
        salida = salida + " " + numeroArreglo[indice];  
    }  
    console.log (salida);  
}
```



```
PROBLEMS  TERMINAL  ...  1: powershell  
PS C:\Proyectos\CFP\2019\JS> node 005-SumarTresArreglos.js  
Ingrese la dimension del arreglo: 6  
17 58 19 49 9 30  
  
16 88 87 24 99 5  
  
La suma de los arreglos es:  
33 146 106 73 108 35  
  
PS C:\Proyectos\CFP\2019\JS>
```

# Ejercicios de Repaso

## *Ejercicio – Personas en una Disco*

- Para tener control de la gente que hay en una disco el gerente quiere saber cuantas personas de diferentes edades han entrado.
- No se han permitido la entrada a menores de 18 ni mayores de 40. Para la carga de los datos se usa la función aleatorio (use la función aleatorio(menorValor,mayorValor), es decir aleatorio(19,40))
- Se sabe que la cantidad total de personas dentro del local es de 270
- Se quiere saber:
  - Cuántas personas son menores de 21 años
  - Cuántas personas mayores o igual a 21 años
  - Cuántas personas en total



# Ejercicios de Repaso

## *Ejercicio – Personas en una Disco*

// Indica la cantidad de menores y de mayores de 21 años que hay en la disco

```
let readlineSync = require('readline-sync');
```

```
let capacidad = 270;
```

```
let personas = new Array(capacidad );
```

```
let menores21 = 0;
```

```
let mayores21 = 0;
```

```
completarBoliche(personas, capacidad)
```

```
menores21= contarMenores(personas, capacidad)
```

```
mostrarPersonas(personas, capacidad)
```

```
console.log ("Los menores de 21 son: ", menores21);
```

```
console.log ("Los mayores de 21 son: ", capacidad - menores21 );
```

```
console.log ("En total hay ", capacidad, " personas");
```



# Ejercicios de Repaso

## *Ejercicio – Personas en una Disco - Errores*

// Completa un arreglo con números enteros ingresados al azar

```
function completarBoliche(personas, capacidad) {  
    let indice;  
  
    for (capacidad=0; capacidad < indice; capacidad++) {  
        personas[indice]=Aleatorio(18,40) ;  
    }  
}
```





# Ejercicios de Repaso

## *Ejercicio – Personas en una Disco - Errores*

// Completa un arreglo con números enteros ingresados al azar

```
function completarBoliche(personas, capacidad) {  
    let indice;  
    for (capacidad=0; capacidad < indice; capacidad++) {  
        personas[indice]=Aleatorio(18,40);  
    }  
}
```

No es indice,  
es capacidad

No es capacidad,  
es indice



# Ejercicios de Repaso

## *Ejercicio – Personas en una Disco*

// Completa un arreglo con números enteros ingresados al azar

```
function completarBoliche(personas, capacidad) {  
    let indice;  
  
    for (indice=0; indice < capacidad; indice++) {  
        personas[indice]=Aleatorio(18,40) ;  
    }  
}
```



# Ejercicios de Repaso

## *Ejercicio – Personas en una Disco - Errores*

// Cuenta la cantidad de menores de 21 que hay en un arreglo

```
function contarMenores(personas, capacidad) {  
    let indice;  
    for (indice=0; indice < capacidad; indice++) {  
        if (personas[indice] > 12) {  
            menores++;  
        }  
    }  
    return menores;  
}
```



# Ejercicios de Repaso

## Ejercicio – Personas en una Disco - Errores

// Cuenta la cantidad de menores de 21 que hay en un arreglo

```
function contarMenores(personas, capacidad) {  
    let indice;  
    for (indice=0; indice < capacidad; indice++) {  
        if (personas[menores] > 12) {  
            menores++;  
        }  
    }  
    return menores;  
}
```

No es > 12,  
es < 21

No es menores,  
es indice

No inicializa  
menores

No define  
menores



# Ejercicios de Repaso

## *Ejercicio – Personas en una Disco*

// Completa un arreglo con números enteros ingresados al azar

```
function completarBoliche(personas, capacidad) {  
    let indice;  
    for (indice=0; indice < capacidad; indice++) {  
        personas[indice]=Aleatorio(18,40) ;  
    }  
}
```

// Cuenta la cantidad de menores de 21 que hay en un arreglo

```
function contarMenores(personas, capacidad) {  
    let menores=0;  
    let indice;  
    for (indice=0; indice < capacidad; indice++) {  
        if (personas[indice] > 21) {  
            menores++;  
        }  
    }  
    return menores;  
}
```



# Ejercicios de Repaso

## *Ejercicio – Personas en una Disco*

// Completa un arreglo con números enteros ingresados al azar

```
function completarBoliche(personas, capacidad) {
    let indice;
    for (indice=0; indice < capacidad; indice++) {
        personas[indice]=Aleatorio(0, capacidad);
    }
}
```

// Cuenta la cantidad de menores de 21 años

```
function contarMenores(personas) {
    let menores=0;
    let indice;
    for (indice=0; indice < personas.length; indice++) {
        if (personas[indice] < 21) {
            menores++;
        }
    }
    return menores;
}
```

```
PROBLEMS  TERMINAL  ...  1: powershell  +  [ ]  [X]  ^  x

PS C:\Proyectos\CFP\2019\JS> node 006-PersonasDisco.js
21 31 22 23 20 34 27 35 26 32 24 27 31 28 24 21 38 27 29 30 33 22 28 18 18 37 22 26 23
32 27 27 29 38 33 26 39 22 35 33 31 30 35 30 34 26 26 26 24 25 30 36 27 32 27 26 23 3
0 33 38 31 37 23 26 34 31 31 20 39 34 22 32 35 37 36 22 20 30 32 23 23 27 31 20 36 20
20 28 22 28 26 23 36 39 21 18 20 36 24 32 38 22 27 24 37 31 36 24 27 29 29 32 39 27 26
19 18 24 33 34 27 18 38 32 18 39 18 36 24 36 29 39 26 26 39 26 37 32 31 22 34 39 19 3
1 23 39 18 37 25 35 29 33 25 26 21 23 36 28 27 31 39 22 22 30 33 19 35 26 29 27 21 37
26 39 18 18 32 30 39 38 33 33 28 30 28 31 19 30 34 21 28 24 26 23 30 32 25 35 21 37 29
35 31 24 20 19 24 27 19 33 26 20 19 20 32 33 28 18 25 21 19 38 35 20 25 23 37 19 18 2
1 38 22 39 32 36 28 19 25 38 23 26 33 35 21 37 22 31 19 27 37 27 18 34 36 26 33 34 39
37 25 27 31 21 28 33 34 23 23 25 37

Los menores de 21 son: 35
Los mayores de 21 son: 235
En total hay 270 personas
PS C:\Proyectos\CFP\2019\JS>
```



# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes*

- Se necesita vender los pasajes de un colectivo. El colectivo tiene 20 filas de 3 butacas cada uno
- Por convención se pone un 0 en el asiento libre y un 1 en el ya vendido. Cuando los pasajes se ponen a la venta están todos los asientos libres
- Cuando una persona quiere comprar se muestra los lugares desocupados y se le asigna un asiento elegido por el pasajero

# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes*

// Se asigna una butaca de un colectivo a un pasajero

```
let readlineSync = require('readline-sync');
```

```
let filas = 20;
```

```
let butacas = 3;
```

```
let asientos = Matriz(filas,butacas); // new Array(filas,butacas);
```

```
completarButacas(asientos, filas, butacas);
```

```
mostrarButacas(asientos, filas, butacas);
```

```
elegirButaca(asientos, filas, butacas);
```

```
mostrarButacas(asientos, filas, butacas);
```





# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes*

// Función que genera un arreglo bi-dimensional

```
function Matriz(filas,butacas) {  
    let matriz = new Array(filas);  
    let indice;  
    for (indice=0; indice < filas; indice++) {  
        matriz[indice] = new Array(butacas);  
    }  
    return matriz;  
}
```



# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes - Errores*

// Se asignan aleatoriamente ocupaciones a las butacas de la matriz

```
function completarButacas(asientos, filas, columnas) {  
    let f, c, filas, columnas;  
    for (f=0; f < columnas; f++) {  
        for (c=0; c <= columnas; c++) {  
            asientos[f][c]= Aleatorio(0,1);  
        }  
    }  
}
```



# Ejercicios de Repaso

## Ejercicio – Vendiendo Pasajes - Errores

// Se asignan aleatoriamente ocupaciones a las butacas de la matriz

**function** completarButacas(asientos, filas, columnas) {

let f, c, **filas, columnas;**

**for** (f=0; f < columnas; f++) {

**for** (c=0; c <= columnas; c++) {

asientos[f][c]= Aleatorio(0,1);

}

}

}

filas y columnas no  
se deben declarar

No es columnas,  
es filas

Es <



# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes*

// Se asignan aleatoriamente ocupaciones a las butacas de la matriz

```
function completarButacas(asientos, filas, columnas) {  
    let f, c;  
    for (f=0; f < filas; f++) {  
        for (c=0; c < columnas; c++) {  
            asientos[f][c]= Aleatorio(0,1);  
        }  
    }  
}
```



# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes*

// Se asignan aleatoriamente ocupaciones a las butacas de la matriz

```
function completarButacas(asientos, filas, columnas) {  
    let f, c;  
    for (f=0; f < filas; f++) {  
        for (c=0; c < columnas; c++) {  
            asientos[f][c] = Aleatorio(0,1);  
        }  
    }  
}
```

// Se pide que digan una posición de butaca para comprar

```
function elegirButaca(asientos, filas, columnas) {  
    let fila, butaca;  
    fila = readlineSync.questionInt( "Indique la fila que desea: ");  
    butaca = readlineSync.questionInt( "Indique la butaca que desea: ");  
    asientos[fila][butaca]=1;  
}
```



# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes*

```
// Muestra las butacas libres y ocupadas de la matriz
function mostrarButacas(asientos, filas, columnas) {
    let f, c;
    let texto="";
    for (c=0; c < columnas; c++)
        cadena=cadena + "B" + (c+1) + " ";
    console.log (cadena);
    for (f=0; f < filas; f++) {
        cadena="Fila "+(f+1)+": ";
        for (c=0; c < columnas; c++) {
            cadena+=String(asientos[f][c]) + " ";
        }
        console.log (cadena);
    }
    console.log ("-----");
}
```



# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes*

```
// Muestra las butacas libres y ocupadas de la matriz
function mostrarButacas(asientos, filas, columnas) {
    let f, c;
    let texto="";
    for (c=0; c < columnas; c++)
        cadena=cadena + "B" + (c+1) + " ";
    console.log (cadena);
    for (f=0; f < filas; f++) {
        cadena="Fila "+(f+1)+": ";
        for (c=0; c < columnas; c++) {
            cadena+=String(asientos[f][c]) + "
        }
        console.log (cadena);
    }
    console.log ("-----");
}
```

```
PROBLEMS  TERMINAL  ...  1: powershell ▼

PS C:\Proyectos\CFP\2019\JS> node 007-VentaPasajes.js
      B1 B2 B3
Fila 1: 0  0  0
Fila 2: 0  0  0
Fila 3: 0  0  0
Fila 4: 0  0  0
Fila 5: 0  0  0
-----
Indique la fila que desea: 3
Indique la butaca que desea: 2
      B1 B2 B3
Fila 1: 0  0  0
Fila 2: 0  0  0
Fila 3: 0  1  0
Fila 4: 0  0  0
Fila 5: 0  0  0
-----
PS C:\Proyectos\CFP\2019\JS>
```



# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes con Reservas*

- Ahora se ha decidido ofrecer la posibilidad de reservar un pasaje
- Cuando una persona quiere hacer una reserva se muestra los lugares libres, el pasajero selecciona su asiento y se marca con un 2. Luego, se le informa al pasajero la ubicación de su asiento
- Cuando la persona va a comprar un pasaje se verifica que ese asiento no esté vendido. Si es así se le informa al pasajero.
- Cuando se va a comprar un pasaje con reserva, se verifica que efectivamente ese asiento esté reservado y se informa.
- Modifique el programa anterior para ofrecer esta nueva funcionalidad



# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes con Reservas*

// Dependiendo de lo que desee el usuario puede realizar una reserva o una compra de pasaje

```
let readlineSync = require('readline-sync');

let filas = 20;
let asiento = 3;
let asientos = Matriz(filas,asiento);
completarAsientos(asientos, filas, asiento);
mostrarButacas(asientos, filas, butacas);
operacion = preguntarOperacion();
if (operacion == 1) {
    mostrarAsientos(asientos, filas, asiento);
    reservarAsiento(asientos);
} else if (operacion == 2) {
    mostrarAsientos(asientos, filas, asiento);
    comprarAsiento(asientos);
} else {
    console.log ("La operacion es invalida");
}
```



# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes con Reservas*

// Pregunta la operación que quiere realizar

```
function preguntarOperacion() {  
    let operacion;  
    console.log("Indique lo que desea hacer: ");  
    console.log("1. Reservar pasaje");  
    console.log("2. Comprar pasaje");  
    operacion = readlineSync.question("");  
    return operacion;  
}
```



# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes con Reservas*

// Pregunta la operación que quiere realizar

```
function preguntarOperacion() {  
    let operacion;  
    console.log("Indique lo que desea hacer: ");  
    console.log("1. Reservar pasaje");  
    console.log("2. Comprar pasaje");  
    operacion = readlineSync.question("");  
    return operacion;  
}
```

// Completa las butacas aleatoriamente con 0, 1 y 2

```
function completarButacas(asientos, filas, columnas) {  
    let f, c;  
    for (f=0; f < filas; f++) {  
        for (c=0; c < columnas; c++) {  
            asientos[f][c] = Aleatorio(0,2);  
        }  
    }  
}
```



# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes con Reservas*

```
// Muestra los asientos del colectivo
function mostrarButacas(asientos, filas, columnas) {
    let f, c;
    let texto="";
    for (c=0; c < columnas; c++)
        cadena=cadena + "B" + (c+1) + " ";
    console.log (cadena);
    for (f=0; f < filas; f++) {
        cadena="Fila "+(f+1)+": ";
        for (c=0; c < columnas; c++) {
            cadena+=String(asientos[f][c]) + " ";
        }
        console.log (cadena);
    }
    console.log ("-----");
}
```



# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes con Reservas*

// Marca un asiento como vendido verificando que no haya estado vendido con anterioridad

```
function comprarAsiento(asientos) {  
  let fila, asiento;  
  fila = readlineSync.question("Indique la fila que desea: ");  
  asiento = readlineSync.question("Indique la asiento que desea: ");  
  if (asientos[fila][asiento] == 2) {  
    console.log("Su asiento ya estaba reservada, ahora puede comprar el pasaje");  
  } else if (asientos[fila][asiento] == 1) {  
    console.log("El asiento seleccionado ya está vendido");  
  } else {  
    console.log("Ud. compro el asiento ", fila, " ", asiento);  
    asientos[fila][asiento]=1;  
  }  
}
```



# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes con Reservas*

// Marca un asiento como reservado verificando que no haya estado vendido ni reservado con anterioridad

```
function reservarAsiento(asientos) {  
  let fila, asiento;  
  fila = readlineSync.question("Indique la fila que desea: ");  
  asiento = readlineSync.question("Indique la asiento que desea: ");  
  if (asientos[fila][asiento] == 1) {  
    console.log ("El asiento seleccionado ya está vendido");  
  } else if (asientos[fila][asiento] == 2) {  
    console.log ("El asiento seleccionado ya está reservado");  
  } else {  
    console.log ("Ud. reservo el asiento ", fila, " ", asiento);  
    asientos[fila][asiento]=2;  
  }  
}
```



# Ejercicios de Repaso

## *Ejercicio – Vendiendo Pasajes con Reservas*

PROBLEMS TERMINAL ... 1: powershell

PS C:\Proyectos\CFP\2019\JS> node 008-VentaReservaPasajes.js

B0 B1 B2

Fila 1: 1 1 0

Fila 2: 0 0 1

Fila 3: 0 1 0

Fila 4: 0 0 1

Fila 5: 0 1 1

-----

Indique lo que desea hacer:

1. Reservar pasaje

2. Comprar pasaje

1

B0 B1 B2

Fila 1: 1 1 0

Fila 2: 0 0 1

Fila 3: 0 1 0

Fila 4: 0 0 1

Fila 5: 0 1 1

-----

Indique la fila que desea: 2

Indique la asiento que desea: 2

Ud. reservo el asiento 1 1

B0 B1 B2

Fila 1: 1 1 0

Fila 2: 0 2 1

Fila 3: 0 1 0

Fila 4: 0 0 1

Fila 5: 0 1 1

-----

PS C:\Proyectos\CFP\2019\JS>

PROBLEMS TERMINAL ... 1: powershell

PS C:\Proyectos\CFP\2019\JS> node 008-VentaReservaPasajes.js

B0 B1 B2

Fila 1: 1 1 0

Fila 2: 1 1 1

Fila 3: 1 1 0

Fila 4: 0 0 1

Fila 5: 0 1 0

-----

Indique lo que desea hacer:

1. Reservar pasaje

2. Comprar pasaje

2

B0 B1 B2

Fila 1: 1 1 0

Fila 2: 1 1 1

Fila 3: 1 1 0

Fila 4: 0 0 1

Fila 5: 0 1 0

-----

Indique la fila que desea: 3

Indique la asiento que desea: 1

El asiento seleccionado ya está vendido

B0 B1 B2

Fila 1: 1 1 0

Fila 2: 1 1 1

Fila 3: 1 1 0

Fila 4: 0 0 1

Fila 5: 0 1 0

-----

PS C:\Proyectos\CFP\2019\JS>

# Repaso Examen

## *Ejercicio – Decir Que Hace!*

```
function metodo1(arr, x, y) {  
  let ax;  
  ax = arr[x];  
  arr[x] = arr[y];  
  arr[y] = ax;  
}
```





# Repaso Examen

## *Ejercicio – Decir Que Hace!*

Este método permite intercambiar los valores en las posiciones “x” e “y” de un arreglo “arr” utilizando una variable auxiliar “ax”

```
function metodo1(arr, x, y) {  
    let ax;  
    ax = arr[x];  
    arr[x] = arr[y];  
    arr[y] = ax;  
}
```



# Repaso Examen

## *Ejercicio – Decir Que Hace!*

```
function metodo2(v, s) {  
  let i, d, aux;  
  i = 0;  
  d = s - 1;  
  while (i < d) {  
    metodo1(v, i, d);  
    i = i + 1;  
    d = d - 1;  
  }  
}
```



# Repaso Examen

## *Ejercicio – Decir Que Hace!*

```
function metodo2(v, s) {  
  let i, d, aux;  
  i = 0;  
  d = s - 1;  
  while (i < d) {  
    metodo1(v, i, d);  
    i = i + 1;  
    d = d - 1;  
  }  
}
```

- Este método invierte los elementos del arreglo “v” de tamaño “s”
- El arreglo se navega con dos índices, denominados “i” y “d”, los cuales permiten analizar el extremo izquierdo y derecho al mismo tiempo
- El índice “i” es incrementado y “d” es decrementado en cada iteración



# Repaso Examen

## *Ejercicio – Decir Que Hace!*

```
function metodo3(mat, x, y) {  
  let a, b;  
  for (a = 0; a < x; a++) {  
    for (b = 0; b < y; b++) {  
      mat[a, b] = (a + 1) * (b + 1);  
    }  
  }  
}
```



# Repaso Examen

## *Ejercicio – Decir Que Hace!*

Si  $x = 4$  e  $y = 4$

```
function metodo3(mat, x, y) {  
  let a, b;  
  for (a = 0; a < x; a++) {  
    for (b = 0; b < y; b++) {  
      mat[a, b] = (a + 1) * (b + 1);  
    }  
  }  
}
```



# Repaso Examen

## *Ejercicio – Decir Que Hace!*

Si  $x = 4$  e  $y = 4$ , “a” va de 0 a 3 y “b” va de 0 a 3

```
function metodo3(mat, x, y) {  
  let a, b;  
  for (a = 0; a < x; a++) {  
    for (b = 0; b < y; b++) {  
      mat[a, b] = (a + 1) * (b + 1);  
    }  
  }  
}
```



# Repaso Examen

## *Ejercicio – Decir Que Hace!*

1   2   3   4

Si  $x = 4$  e  $y = 4$ , “a” va de 0 a 3 y “b” va de 0 a 3  
 $\text{mat}[0,0]=1$   $\text{mat}[0,1]=2$   $\text{mat}[0,2]=3$   $\text{mat}[0,3]=4$

```
function metodo3(mat, x, y) {  
  let a, b;  
  for (a = 0; a < x; a++) {  
    for (b = 0; b < y; b++) {  
      mat[a, b] = (a + 1) * (b + 1);  
    }  
  }  
}
```



# Repaso Examen

## *Ejercicio – Decir Que Hace!*

1	2	3	4	Si $x = 4$ e $y = 4$ , “a” va de 0 a 3 y “b” va de 0 a 3
2	4	6	8	$\text{mat}[0,0]=1$ $\text{mat}[0,1]=2$ $\text{mat}[0,2]=3$ $\text{mat}[0,3]=4$
				$\text{mat}[1,0]=2$ $\text{mat}[1,1]=4$ $\text{mat}[1,2]=6$ $\text{mat}[1,3]=8$

```
function metodo3(mat, x, y) {  
    let a, b;  
    for (a = 0; a < x; a++) {  
        for (b = 0; b < y; b++) {  
            mat[a, b] = (a + 1) * (b + 1);  
        }  
    }  
}
```





# Repaso Examen

## *Ejercicio – Decir Que Hace!*

1	2	3	4	Si $x = 4$ e $y = 4$ , “a” va de 0 a 3 y “b” va de 0 a 3
2	4	6	8	$\text{mat}[0,0]=1$ $\text{mat}[0,1]=2$ $\text{mat}[0,2]=3$ $\text{mat}[0,3]=4$
3	6	9	12	$\text{mat}[1,0]=2$ $\text{mat}[1,1]=4$ $\text{mat}[1,2]=6$ $\text{mat}[1,3]=8$
				$\text{mat}[2,0]=2$ $\text{mat}[2,1]=6$ $\text{mat}[2,2]=9$ $\text{mat}[2,3]=12$

```
function metodo3(mat, x, y) {  
    let a, b;  
    for (a = 0; a < x; a++) {  
        for (b = 0; b < y; b++) {  
            mat[a, b] = (a + 1) * (b + 1);  
        }  
    }  
}
```



# Repaso Examen

## *Ejercicio – Decir Que Hace!*

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16

Si  $x = 4$  e  $y = 4$ , “a” va de 0 a 3 y “b” va de 0 a 3

$\text{mat}[0,0]=1$   $\text{mat}[0,1]=2$   $\text{mat}[0,2]=3$   $\text{mat}[0,3]=4$

$\text{mat}[1,0]=2$   $\text{mat}[1,1]=4$   $\text{mat}[1,2]=6$   $\text{mat}[1,3]=8$

$\text{mat}[2,0]=2$   $\text{mat}[2,1]=6$   $\text{mat}[2,2]=9$   $\text{mat}[2,3]=12$

```
function metodo3(mat, x, y) {  
    let a, b;  
    for (a = 0; a < x; a++) {  
        for (b = 0; b < y; b++) {  
            mat[a, b] = (a + 1) * (b + 1);  
        }  
    }  
}
```



# Repaso Examen

## *Ejercicio – Decir Que Hace!*

Este método inicializa una matriz “mat” de dimensión  $x \times y$  con las tablas de multiplicar

El contenido de la matriz representa la multiplicación según los índices + 1, por ejemplo:  $\text{mat}[3,4] = 4 \times 5$  o  $\text{mat}[1,1] = 2 \times 2$

```
function metodo3(mat, x, y) {  
    let a, b;  
    for (a = 0; a < x; a++) {  
        for (b = 0; b < y; b++) {  
            mat[a, b] = (a + 1) * (b + 1);  
        }  
    }  
}
```

