

Técnicas de Programación

CFL

**Programador
full-stack**

Modularización y Métodos (Repaso)

Métodos

Repaso

- **Agrupan** 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

Repaso

- 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;  
  let s="";  
  for (x=1; x<=40; x++) {  
    s=s+"-";  
  }  
  console.log(s);  
}
```

Métodos con Retorno

Sintaxis

```
function nombre_del_metodo(argumento_1,argumento_2,... ) {  
  let retorno;  
  acción 1  
  acción 2  
  ...  
  acción n  
  
  return retorno;  
}
```

Técnicas de Programación

CFL

**Programador
full-stack**

Ámbito de las Variables (Concepto)

Naming

- Variables:
 - se nombran con sustantivos
- Funciones:
 - Comienzan con verbos
 - En el caso de funciones booleanas, deben comenzar con "is", ej: isValid()
- **Usar nombres descriptivos**
 - nunca son demasiado largos!
- Evitar nombres sin significado como "aux" y "temp"

Técnicas de Programación

CFL

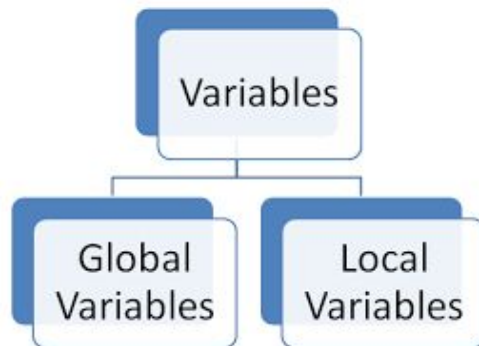
**Programador
full-stack**

Ámbito de las Variables (Concepto)

Ámbito de las Variables

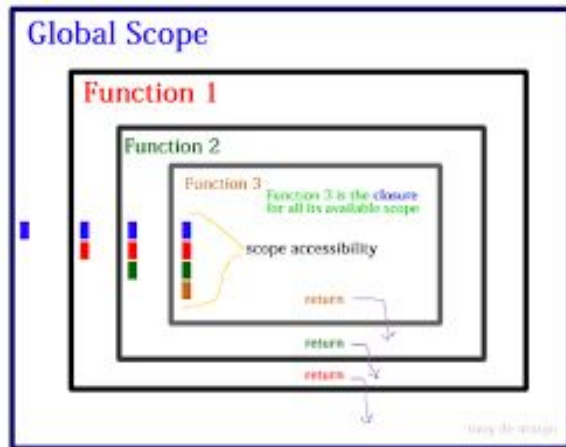
Al utilizar funciones se establece un límite para el alcance de las variables

- **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 | zarse en cualquier método



Ámbito de las Variables

- Se debe intentar crear métodos con variables locales y pocos parámetros para favorecer la reutilización y el mantenimiento del software



Ámbito de las Variables

Ejemplos

```
let mensaje = 'Hola Global!';
```

```
ambitoVariables();
```

```
function ambitoVariables() {
```

```
    let mensaje;
```

```
    mensaje = 'Hola Mundo!!';
```

```
    console.log(mensaje);
```

```
}
```

Ámbito de las Variables

Ejemplos

```
let mensaje = 'Hola Global!';
```

```
ambitoVariables();
```

```
function ambitoVariables() {
```

```
  let mensaje;
```

```
  mensaje = 'Hola Mundo!!';
```

```
}
```

La variable local
esconde la global

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Proyectos\CFP\2019\JS> node 009-AmbitoVariables.js  
Hola Mundo!!  
PS C:\Proyectos\CFP\2019\JS> 
```

Ámbito de las Variables

Ejemplos

```
ambitoVariables();
```

```
function ambitoVariables() {  
  let mensaje;  
  mensaje = 'Hola Mundo!!';  
  console.log(mensaje);  
}
```

```
ambitoVariables();
```

```
function ambitoVariables() {  
  leeVariable();  
}
```

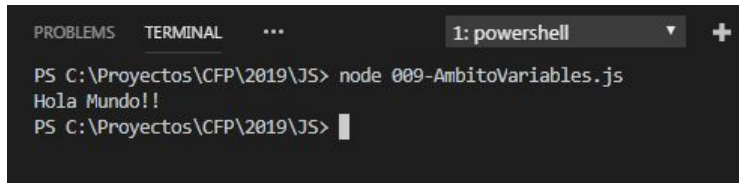
```
function leeVariable() {  
  let mensaje;  
  mensaje = 'Hola Mundo!!';  
  console.log(mensaje);  
}
```

Ámbito de las Variables

Ejemplos

```
ambitoVariables();
```

```
function ambitoVariables() {  
  let mensaje;  
  mensaje = 'Hola Mundo!!';  
  console.log(mensaje);  
}
```



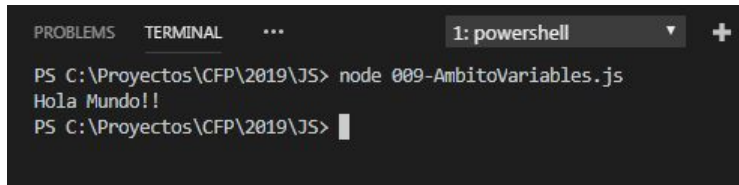
A terminal window with a dark background and light text. The title bar shows '1: powershell'. The terminal content shows the command 'node 009-AmbitoVariables.js' being executed, which outputs 'Hola Mundo!!'.

```
PROBLEMS  TERMINAL  ...  1: powershell  +  
PS C:\Proyectos\CFP\2019\JS> node 009-AmbitoVariables.js  
Hola Mundo!!  
PS C:\Proyectos\CFP\2019\JS> |
```

```
ambitoVariables();
```

```
function ambitoVariables() {  
  leeVariable();  
}
```

```
function leeVariable() {  
  let mensaje;  
  mensaje = 'Hola Mundo!!';  
  console.log(mensaje);  
}
```



A terminal window with a dark background and light text. The title bar shows '1: powershell'. The terminal content shows the command 'node 009-AmbitoVariables.js' being executed, which outputs 'Hola Mundo!!'.

```
PROBLEMS  TERMINAL  ...  1: powershell  +  
PS C:\Proyectos\CFP\2019\JS> node 009-AmbitoVariables.js  
Hola Mundo!!  
PS C:\Proyectos\CFP\2019\JS> |
```

Ámbito de las Variables

Ejemplos

```
let mensaje;  
ambitoVariables();
```

```
function ambitoVariables() {  
    mensaje = 'Hola Mundo!!';  
    console.log(mensaje);  
}
```

```
let mensaje;  
ambitoVariables();
```

```
function ambitoVariables() {  
    leeVariable();  
}  
function leeVariable() {  
    mensaje = 'Hola Mundo!!';  
    console.log(mensaje);  
}
```

Ámbito de las Variables

Ejemplos

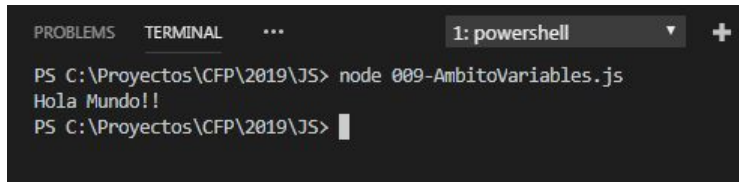
```
let mensaje;  
ambitoVariables();
```

```
function ambitoVariables() {  
    mensaje = 'Hola Mundo!!';  
    console.log(mensaje);  
}
```

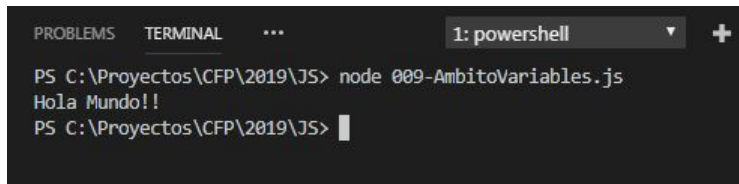
```
let mensaje;  
ambitoVariables();
```

```
function ambitoVariables() {  
    leeVariable();  
}
```

```
function leeVariable() {  
    mensaje = 'Hola Mundo!!';  
    console.log(mensaje);  
}
```



```
PROBLEMS  TERMINAL  ...  1: powershell  +  
  
PS C:\Proyectos\CFP\2019\JS> node 009-AmbitoVariables.js  
Hola Mundo!!  
PS C:\Proyectos\CFP\2019\JS> |
```



```
PROBLEMS  TERMINAL  ...  1: powershell  +  
  
PS C:\Proyectos\CFP\2019\JS> node 009-AmbitoVariables.js  
Hola Mundo!!  
PS C:\Proyectos\CFP\2019\JS> |
```

Ámbito de las Variables

Ejemplos

```
let mensaje;  
ambitoVariables();
```

```
function ambitoVariables() {  
    mensaje = 'Hola Mundo!!';  
    leeVariable();  
}  
function leeVariable() {  
    console.log(mensaje);  
}
```

```
let mensaje;  
ambitoVariables();
```

```
function ambitoVariables() {  
    leeVariable();  
    mensaje = 'Hola Mundo!!';  
}
```

```
function leeVariable() {
```


Ámbito de las Variables

Ejemplos

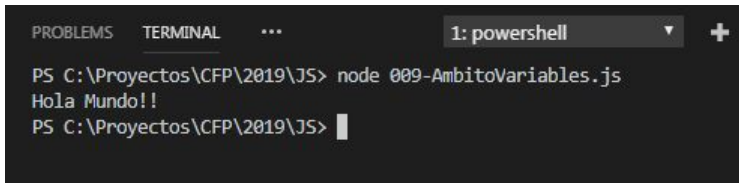
```
let mensaje;  
ambitoVariables();
```

```
function ambitoVariables() {  
    mensaje = 'Hola Mundo!!';  
    leeVariable();  
}  
function leeVariable() {  
    console.log(mensaje);  
}
```

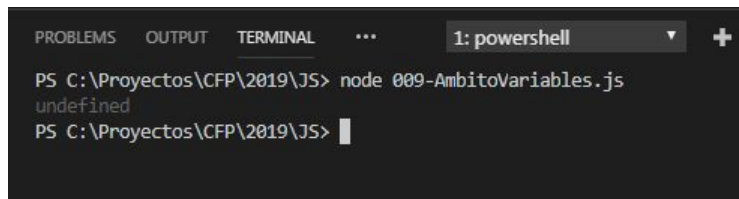
```
let mensaje;  
ambitoVariables();
```

```
function ambitoVariables() {  
    leeVariable();  
    mensaje = 'Hola Mundo!!';  
}
```

```
function leeVariable() {
```



```
PROBLEMS  TERMINAL  ...  1: powershell  +  
PS C:\Proyectos\CFP\2019\JS> node 009-AmbitoVariables.js  
Hola Mundo!!  
PS C:\Proyectos\CFP\2019\JS>
```



```
PROBLEMS  OUTPUT  TERMINAL  ...  1: powershell  +  
PS C:\Proyectos\CFP\2019\JS> node 009-AmbitoVariables.js  
undefined  
PS C:\Proyectos\CFP\2019\JS>
```

Técnicas de Programación

CFL

**Programador
full-stack**

Buenas Prácticas de Programación (Concepto)

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



Técnicas de Programación

CFL

**Programador
full-stack**

Arreglos (Conceptos)

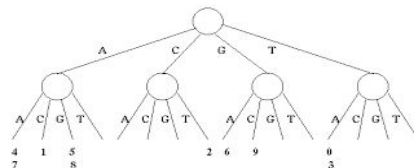
Estructuras de Datos

Forma particular de organizar datos

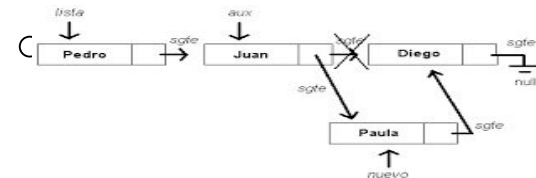


- Estructuras que permiten **COLECCIONAR** elementos
- Estructuras

- GUARDARLOS
- RECORRERLOS
- MANIPULARLOS



- **LISTAS**
- **COLAS**
- **PILAS**



- Operaciones básicas

- **COLOCAR**

Estructuras de Datos - Arreglos

Construya un algoritmo que según el número de mes muestre el nombre de dicho mes

¿Cómo se puede hacer?



Estructuras de Datos – Arreglos

Ejercicio - Identificación Mes - Código

//Algoritmo Mes

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

```
let nroMes = readlineSync.question("Indique el número de mes que le interesa");
```

```
switch (nroMes) {
```

```
  case 1: console.log("El mes es Enero"); break;
```

```
  case 2: console.log("El mes es Febrero"); break;
```

```
  case 3: console.log("El mes es Marzo"); break;
```

```
  case 4: console.log("El mes es Abril"); break;
```

```
  case 5: console.log("El mes es Mayo"); break;
```

```
  case 6: console.log("El mes es Junio"); break;
```

```
  case 7: console.log("El mes es Julio"); break;
```

```
  case 8: console.log("El mes es Agosto"); break;
```

```
  case 9: console.log("El mes es Septiembre"); break;
```

```
  case 10: console.log("El mes es Octubre"); break;
```

```
  case 11: console.log("El mes es Noviembre"); break;
```

1. ENERO	7. JULIO
2. FEBRERO	8. AGOSTO
3. MARZO	9. SEPTIEMBRE
4. ABRIL	10. OCTUBRE
5. MAYO	11. NOVIEMBRE
6. JUNIO	12. DICIEMBRE

Estructuras de Datos – Arreglos

Otras Necesidades

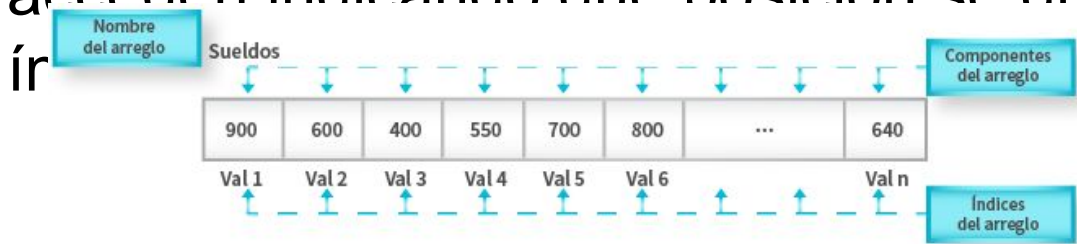
- ¿Qué pasa si en lugar de meses fueran clientes y números de clientes?
- A medida que tengo más clientes tengo que programar más Segun / Si... imposibles en aplicaciones reales



Estructuras de Datos

Arreglos/Listas/Vectores

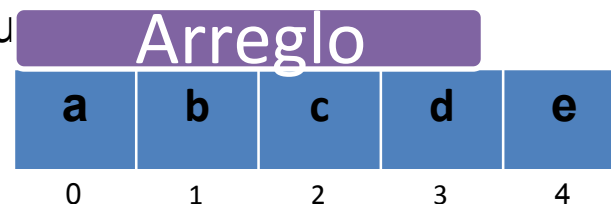
- Los arreglos son estructuras de datos homogéneas (todos sus datos son del mismo tipo)
- Permiten almacenar un determinado número de datos
- Tiene muchos elementos, y a cada uno de ellos se acceden indicando que posición se quiere usar (un



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 primer elemento más 1
- Propiedades:
 - ELEMENTO o ITEM: a, b, c, d, e
 - LONGITUD: 5



Longitud = Length = 5

Estructuras de Datos

Definición de Arreglos

let <identificador> = **new Array** (<maxI>);

Ejemplo:

let arregloClientes = **new Array** (30);

- Esta instrucción define un arreglo con el nombre indicado en <identificador> y 1 dimensión
- El parámetro indica el valor máximo de elementos.
- La cantidad de dimensiones debe ser una, y la máxima cantidad de elementos debe ser una expresión numérica positiva
- Mas adelante veremos la manera de implementar arreglos de mas de una dimensión.

Estructuras de Datos – Arreglos

Ejercicio - Identificación Mes

- Modifique el código del Ejercicio Identificación mes utilizando arreglos

Longitud = Length= 12

Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio
0	1	2	3	4	5	6



Estructuras de Datos – Arreglos

Ejercicio - Identificación Mes – Código

//Algoritmo Mes

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

```
let arregloMes = new Array (12);
```

```
arregloMes[0] = "Enero";
```

```
arregloMes[1] = "Febrero";
```

```
arregloMes[2] = "Marzo";
```

```
arregloMes[3] = "Abril";
```

```
arregloMes[4] = "Mayo";
```

```
arregloMes[5] = "Junio";
```

```
arregloMes[6] = "Julio";
```

```
arregloMes[7] = "Agosto";
```

```
arregloMes[8] = "Septiembre";
```

```
arregloMes[9] = "Octubre";
```

```
arregloMes[10] = "Noviembre";
```

Estructuras de Datos – Arreglos

Ejercicio - Identificación Mes – Código

//Algoritmo Mes

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

```
let arregloMes = new Array (12) ;
```

```
arregloMes[0] = "Enero";
```

```
arregloMes[1] = "Febrero";
```

```
arregloMes[2] = "Marzo";
```

```
arregloMes[3] = "Abril";
```

```
arregloMes[4] = "Mayo";
```

```
arregloMes[5] = "Junio";
```

```
arregloMes[6] = "Julio";
```

```
arregloMes[7] = "Agosto";
```

```
arregloMes[8] = "Septiembre";
```

```
arregloMes[9] = "Octubre";
```

```
arregloMes[10] = "Noviembre";
```

Recuerde que al ser el arreglo en base 0
hay que restar 1 al índice

Estructuras de Datos – Arreglos

Ejercicio – Arreglo de Números

- Construya un algoritmo que tenga un arreglo de números y se los muestre al usuario
- El arreglo debe ser llamado num
- El arreglo num debe contener los siguientes datos: 20, 14, 8, 0, 5, 19 y 24.
- Mostrar los valores resultantes del arreglo

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/indice del num entonces van desde

	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=0;

num[0] = 20;

num[1] = 14;

num[2] = 8;

num[3] = 0;

num[4] = 5;

num[5] = 19;

num[6] = 24;

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=0;
```

```
num[0] = 20;
```

```
num[1] = 14;
```

```
num[2] = 8;
```

```
num[3] = 0;
```

```
num[4] = 5;
```

```
num[5] = 19;
```

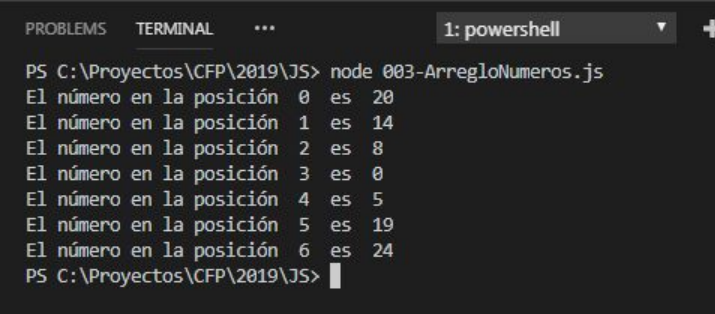
```
num[6] = 24;
```

```
while (indice < 7) {
```

```
    console.log ("El número en la posición ", indice, " es ", num[indice]);
```

```
    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>
```

Estructuras de Datos – Arreglos

Ejercicio – Números Deseados

- Construya un algoritmo que tenga un arreglo de dimensión 5 y llénelo con los números que el usuario desee
- Muestre los números del arreglo al usuario

Estructuras de Datos – Arreglos

Ejercicio – Números Deseados - Código

//Algoritmo NumerosDeseados

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

```
let nroDeseadoArreglo = new Array (5);
```

```
let nro, indice;
```

```
for (indice = 0; indice < 5; indice++) {
```

```
    nro = readlineSync.questionInt("Indique el numero que desea incorporar en la posicion "+indice+": ");
```

```
    nroDeseadoArreglo[indice] = nro;
```

```
}
```

```
for (indice = 0; indice < 5; indice++) {
```

```
    console.log("El numero en la posicion ", indice, " es ", nroDeseadoArreglo[indice]);
```

```
}
```

Estructuras de Datos – Arreglos

Ejercicio – Números Deseados - Código

//Algoritmo NumerosDeseados

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

```
let nroDeseadoArreglo = new Array (5);
```

```
let nro, indice;
```

```
for (indice = 0; indice < 5; indice++) {
```

```
    nro = readlineSync.questionInt("Indique el numero que desea incorporar en la posicion "+indice+": ");
```

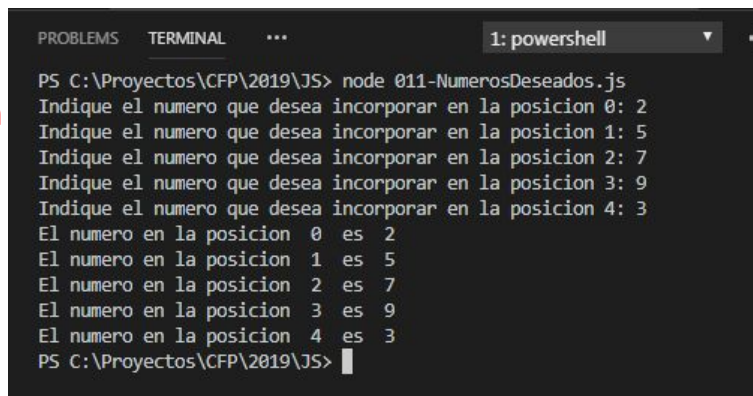
```
    nroDeseadoArreglo[indice] = nro;
```

```
}
```

```
for (indice = 0; indice < 5; indice++) {
```

```
    console.log("El numero en la posicion
```

```
}
```



```
PROBLEMS  TERMINAL  ...  1: powershell
PS C:\Proyectos\CFP\2019\JS> node 011-NumerosDeseados.js
Indique el numero que desea incorporar en la posicion 0: 2
Indique el numero que desea incorporar en la posicion 1: 5
Indique el numero que desea incorporar en la posicion 2: 7
Indique el numero que desea incorporar en la posicion 3: 9
Indique el numero que desea incorporar en la posicion 4: 3
El numero en la posicion 0 es 2
El numero en la posicion 1 es 5
El numero en la posicion 2 es 7
El numero en la posicion 3 es 9
El numero en la posicion 4 es 3
PS C:\Proyectos\CFP\2019\JS>
```

Estructuras de Datos – Arreglos

Ejercicio – Nombres Deseados

- Construya un algoritmo que tenga un arreglo de dimensión deseada por el usuario y llénelo con los nombres que el usuario desee
- Crear un arreglo de las posiciones que desee el usuario y llenarlo con nombres de personas

Estructuras de Datos – Arreglos

Ejercicio – Nombres Deseados - Código

```
//Algoritmo NombresPersonas
```

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

```
let dimensionArreglo = readlineSync.questionInt("Ingrese la dimensión del arreglo: ");
```

```
let nombrePersonas = new Array (dimensionArreglo);
```

```
let nombre , indice;
```

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

```
    nombre = readlineSync.question("Ingrese el nombre que quiere poner en el lugar "+indice+": ");
```

```
    nombrePersonas[indice] = nombre;
```

```
}
```

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

```
    console.log("La persona que ingreso en la posición ", indice, " es: ", nombrePersonas[indice]);
```

```
}
```


Estructuras de Datos – Arreglos

Ejercicio – Nombres Deseados - Código

```
//Algoritmo NombresPersonas
```

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

```
let dimensionArreglo = readlineSync.questionInt("Ingrese la dimensión del arreglo: ");
```

```
let nombrePersonas = new Array (dimensionArreglo);
```

```
let nombre , indice;
```

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

```
    nombre = readlineSync.question("Ingrese el nombre que quiere poner en el lugar "+indice+": ");
```

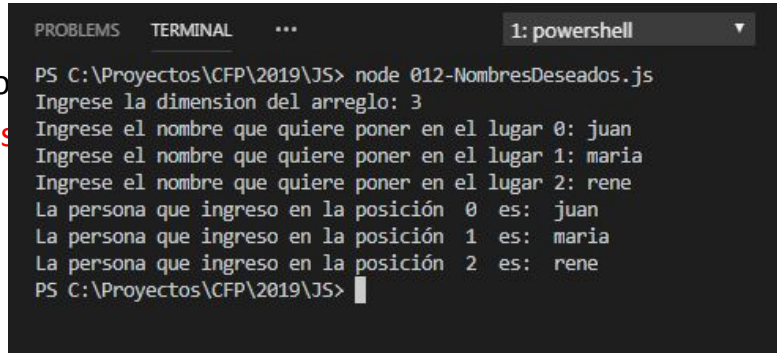
```
    nombrePersonas[indice] = nombre;
```

```
}
```

```
for (indice = 0; indice < dimensionArreglo
```

```
    console.log("La persona que ingreso en la posición "+indice));
```

```
}
```



```
PROBLEMS  TERMINAL  ...  1: powershell
PS C:\Proyectos\CFP\2019\JS> node 012-NombresDeseados.js
Ingrese la dimension del arreglo: 3
Ingrese el nombre que quiere poner en el lugar 0: juan
Ingrese el nombre que quiere poner en el lugar 1: maria
Ingrese el nombre que quiere poner en el lugar 2: rene
La persona que ingreso en la posición 0 es: juan
La persona que ingreso en la posición 1 es: maria
La persona que ingreso en la posición 2 es: rene
PS C:\Proyectos\CFP\2019\JS>
```

Estructuras de Datos – Arreglos

Ejercicio – Dos Arreglos

- Construya un algoritmo que tenga dos arreglos uno que almacene 2 nombres y otro que almacene 3 números ambos ingresados por el usuario.
- Al final debe imprimir los valores por consola.

Estructuras de Datos – Arreglos

Ejercicio – Dos Arreglos - Código

//Algoritmo DosArreglos

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

```
let arregloNombres = new Array (2);
```

```
let arregloNumeros = new Array (3);
```

```
let nombre, numero, indice;
```

```
for (indice = 0; indice < 2; indice++) {
```

```
    nombre = readlineSync.question("Ingrese el nombre de la posicion "+indice+": ");
```

```
    arregloNombres[indice] = nombre;
```

```
}
```

```
for (indice = 0; indice < 3; indice++) {
```

```
    numero = readlineSync.questionInt("Ingrese el numero de la posicion "+indice+": ");
```

```
    arregloNumeros[indice] = numero;
```

```
}
```

```
for (indice = 0; indice < 2; indice++) {
```

```
    console.log("El nombre en la posición ", indice, " es: ", arregloNombres[indice]);
```

Estructuras de Datos – Arreglos

Ejercicio – Dos Arreglos - Código

```
//Algoritmo DosArreglos
```

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

```
let arregloNombres = new Array (3);
```

```
let arregloNumeros = new Array (2);
```

```
let nombre, numero, indice;
```

```
for (indice = 0; indice < 3; indice++) {
```

```
    nombre = readlineSync.question("Ingrese el nombre de la posicion "+indice+": ");
```

```
    arregloNombres[indice] = nombre;
```

```
}
```

```
for (indice = 0; indice < 2; indice++) {
```

```
    numero = readlineSync.questionInt("Ingrese el numero de la posicion "+indice+": ");
```

```
    arregloNumeros[indice] = numero;
```

```
}
```

```
for (indice = 0; indice < 3; indice++) {
```

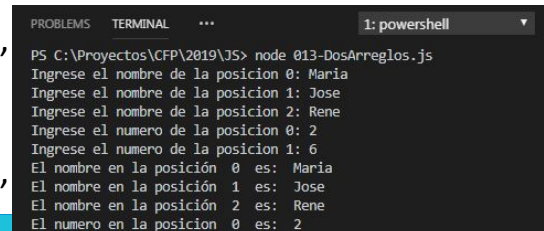
```
    console.log("El nombre en la posición ", indice, " es: ",
```

```
}
```

```
for (indice = 0; indice < 2; indice++) {
```

```
    console.log("El numero en la posicion ", indice, " es: ",
```

```
}
```



```
PROBLEMS  TERMINAL  ...  1: powershell
PS C:\Proyectos\CFP\2019\JS> node 013-DosArreglos.js
Ingrese el nombre de la posicion 0: Maria
Ingrese el nombre de la posicion 1: Jose
Ingrese el nombre de la posicion 2: Rene
Ingrese el numero de la posicion 0: 2
Ingrese el numero de la posicion 1: 6
El nombre en la posición 0 es: Maria
El nombre en la posición 1 es: Jose
El nombre en la posición 2 es: Rene
El numero en la posición 0 es: 2
```

Estructuras de Datos – Arreglos

Ejercicio – Suma Elementos Arreglo

- Construya un algoritmo que sume todos los elementos de un arreglo de tamaño N
- La dimensión del arreglo es ingresada por el usuario
- Los elementos (números) del arreglo son ingresados por el usuario

Estructuras de Datos – Arreglos

Ejercicio – Suma Elementos Arreglo - Código

//Algoritmo SumaArreglo

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

let dimensionArreglo = readlineSync.questionInt("Ingrese la dimension del arreglo: ");
let arreglo = new Array (dimensionArreglo);
let numeroArreglo, indice, resultado;
for (indice = 0; indice < dimensionArreglo; indice++) {
    numeroArreglo = readlineSync.questionInt("Indique el nro que va en la posicion "+indice+": ");
    arreglo[indice] = numeroArreglo;
}
resultado = 0;
for (indice = 0; indice < dimensionArreglo; indice++) {
    resultado += arreglo[indice];
}
for (indice = 0; indice < dimensionArreglo; indice++) {
    console.log("El numero en la posicion ", indice, " es: ", arreglo[indice]);
}
```

Estructuras de Datos – Arreglos

Ejercicio – Suma Elementos Arreglo - Código

```
//Algoritmo SumaArreglo
```

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

```
let dimensionArreglo = readlineSync.questionInt("Ingrese la dimension del arreglo: ");
```

```
let arreglo = new Array (dimensionArreglo);
```

```
let numeroArreglo, indice, resultado;
```

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

```
    numeroArreglo = readlineSync.questionInt("Indique el nro que va en la posicion ");
```

```
    arreglo[indice] = numeroArreglo;
```

```
}
```

```
resultado = 0;
```

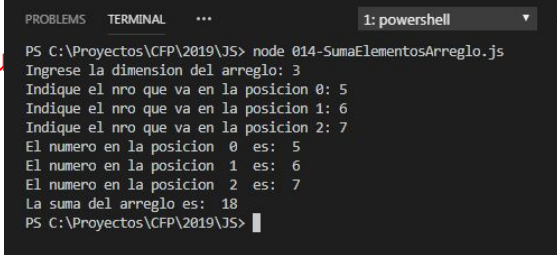
```
for (indice = 0; indice < dimensionArreglo; indice++) {
```

```
    resultado += arreglo[indice];
```

```
}
```

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

```
    console.log("El numero en la posicion ", indice, " es: ", arreglo[indice]);
```



```
PS C:\Proyectos\CFP\2019\JS> node 014-SumaElementosArreglo.js
Ingrese la dimension del arreglo: 3
Indique el nro que va en la posicion 0: 5
Indique el nro que va en la posicion 1: 6
Indique el nro que va en la posicion 2: 7
El numero en la posicion 0 es: 5
El numero en la posicion 1 es: 6
El numero en la posicion 2 es: 7
La suma del arreglo es: 18
PS C:\Proyectos\CFP\2019\JS>
```

Estructuras de Datos – Arreglos

Ejercicio – Completar Arreglo

- Llenar un array de 10 posiciones con números aleatorios entre 0 y 99
- Para obtener números aleatorios crear una función Azar, que se base en las funciones disponibles en el paquete Math:
 - Math.random() devuelve un nro al azar entre 0 y 1.

Estructuras de Datos – Arreglos

Ejercicio – Completar Arreglo - Código

```
//Algoritmo CompletarArreglo
```

```
let arregloCompletar = new Array (10);
```

```
let indice;
```

```
for (indice = 0; indice < 10; indice++) {
```

```
    arregloCompletar[indice] = Azar(100);
```

```
}
```

```
for (indice = 0; indice < 10; indice++) {
```

```
    console.log ("El numero en la posicion ", indice, " es: ", arregloCompletar[indice]);
```

```
}
```

```
function Azar (tope) {
```

```
    return Math.floor(Math.random()*tope);
```

```
};
```

Estructuras de Datos – Arreglos

Ejercicio – Completar Arreglo - Código

```
//Algoritmo CompletarArreglo
```

```
let arregloCompletar = new Array (10);
```

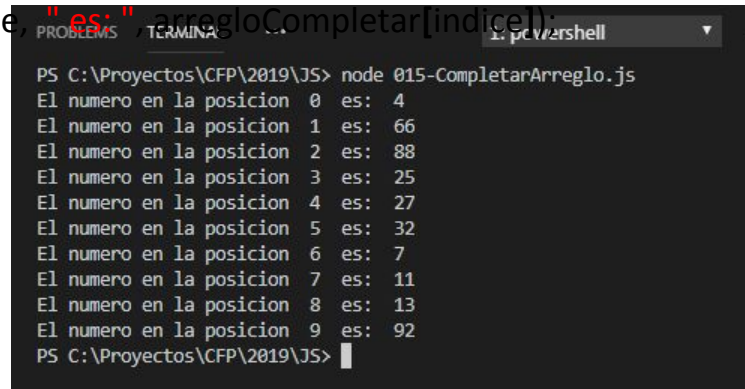
```
let indice;
```

```
for (indice = 0; indice < 10; indice++) {  
    arregloCompletar[indice] = Azar(100);
```

```
}
```

```
for (indice = 0; indice < 10; indice++) {  
    console.log ("El numero en la posicion ", indice, " es: ", arregloCompletar[indice]);
```

```
function Azar (tope) {  
    return Math.floor(Math.random()*tope);  
};
```



```
PS C:\Proyectos\CFP\2019\JS> node 015-CompletarArreglo.js  
El numero en la posicion 0 es: 4  
El numero en la posicion 1 es: 66  
El numero en la posicion 2 es: 88  
El numero en la posicion 3 es: 25  
El numero en la posicion 4 es: 27  
El numero en la posicion 5 es: 32  
El numero en la posicion 6 es: 7  
El numero en la posicion 7 es: 11  
El numero en la posicion 8 es: 13  
El numero en la posicion 9 es: 92  
PS C:\Proyectos\CFP\2019\JS>
```

Técnicas de Programación

CFL

**Programador
full-stack**

Arreglos (Ejercicios)

Estructuras de Datos

Sumar Dos Arreglos

- Sumar los elementos de cada una de las posiciones de dos arreglos y guardar el resultado en otro arreglo
- El arreglo tiene dimensión 6 y los números de los dos vectores los carga el usuario

Ejemplo:

v1 = 1, 3, 7, 9, 9, 5
 v2 = 6, 9, 2, 5, 9, 4
 vSuma = 7, 12, 9, 14, 18, 9

$$\mathbf{A + B =}$$

$$\mathbf{<(a_1 + a_2),(b_1 + b_2),(c_1 + c_2)>}$$

ex.

$$\mathbf{A = <5, 9, -10> \quad B = <17, -3, -2>}$$

$$\mathbf{A+B = <(5+17),(9+(-3)),((-10)+(-2))>}$$

$$\mathbf{= <22, 6, -12>}$$

Estructuras de Datos

Invertir Arreglo

- **Almacene** en un arreglo de tamaño **N** los números ingresados por el usuario
- La **dimensión N** también es ingresada por el usuario
- **Muestre** los números del arreglo pero de

Ejemplo:

v = 1, 3, 7, 9, 9, 5
La salida es: 5, 9, 9, 7, 3, 1



Estructuras de Datos

Tipos de Números en Arreglo



- Almacene en un arreglo de dimensión N números (la cantidad es ingresada por el usuario)
- Muestre cuántos números son positivos, cuántos son negativos y cuántos ceros hay

Ejemplo:

v = 0, -7, -9, 1, 0, 0

La salida es: 1 positivos, 2 negativos y 2 ceros

