

**Resolución a problemas algorítmicos aplicando
estructuras de almacenamiento.**

GA3-220501093-AA3-EV02



Isidro J Gallardo Navarro

Ficha:3070299

2025

**Tecnología en Análisis y Desarrollo de
Software.**

ADSO

INTRODUCCIÓN

El presente documento constituye la materialización de la evidencia de desempeño GA3-220501093-AA3-EV02: Resolución a problemas algorítmicos aplicando estructuras de almacenamiento. Esta evidencia tiene como propósito fundamental aplicar todos los conocimientos adquiridos a lo largo del componente formativo para dar solución efectiva a una serie de problemas algorítmicos propuestos, demostrando el dominio de las competencias técnicas en programación.

Las soluciones desarrolladas han sido implementadas utilizando el lenguaje JavaScript, empleando exclusivamente las estructuras de control básicas requeridas (condicionales if-else, bucles for) y estructuras de almacenamiento como vectores, sin recurrir a funciones avanzadas o métodos complejos, garantizando así una aproximación purista a los fundamentos de la programación.

El alcance del desarrollo aborda múltiples problemas que requieren la implementación rigurosa de lógica de programación y el manejo eficiente de datos estructurados. Las tareas principales logradas incluyen: el cálculo de área de figuras planas (triángulo, rectángulo, cuadrado y círculo) con validación de entrada de datos; el análisis estadístico de edades almacenadas en un vector de 10 elementos con validación de rangos específicos; la mezcla y ordenación de vectores de números enteros implementando algoritmos de ordenamiento eficientes; y la gestión de registro de datos personales y gustos musicales mediante estructuras de almacenamiento paralelas que permiten manejar información compleja de hasta 6 personas con sus respectivas canciones favoritas.

Cada solución algorítmica ha sido diseñada para demostrar no solo la correcta aplicación de las estructuras de datos fundamentales, sino también la capacidad de resolver problemas reales mediante la implementación de algoritmos eficientes y la validación adecuada de datos de entrada.

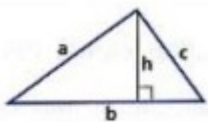
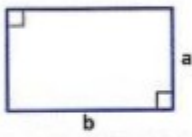
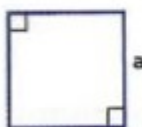

En esta evidencia introductoria al lenguaje Javascript desarrollare los cuatro ejercicios propuestos en la guía de aprendizaje #3

1. Desarrollar un programa que permita calcular el área o perímetro de algunas figuras planas según la siguiente tabla:

Tabla 1 Área y perímetro de figuras planas

Triángulo, Rectángulo, Cuadrado, Círculo

Para este punto de la evidencia desarrollare un código que halle el área de figuras geométricas

Figura	Perímetro	Área
	$a + b + c$	$\frac{b \times h}{2}$
	$2 \times (b + a)$	$b \times a$
	$4 \times a$	a^2
	$2 \times \pi \times r$	$\pi \times r^2$

El código a desarrollar está centrado en hallar el área de las figuras planas señaladas en las anteriores imágenes con una variable let la cual selecciona la figura en cuestión, el código es sencillo, los parámetros de las figuras deben ser integrados en el código así como la selección de la figura en la variable **"let figura"**

CODE:

```
const PI = 3.14159;

console.log("=== CALCULADORA DE ÁREAS DE FIGURAS  
GEOMÉTRICAS ===");

let figura = 1;

console.log("Figura seleccionada: " + figura);

if (figura === 1) {

console.log("Calculando área del TRIÁNGULO");
let base = 10;
let altura = 8;
console.log("Base: " + base);
console.log("Altura: " + altura);
let area = (base * altura) / 2;
console.log("El área del triángulo es: " + area);
} else if (figura === 2) {
console.log("Calculando área del RECTÁNGULO");
let largo = 12;
let ancho = 5;
console.log("Largo: " + largo);
console.log("Ancho: " + ancho);
let area = largo * ancho;
console.log("El área del rectángulo es: " + area);
} else if (figura === 3) {
console.log("Calculando área del CUADRADO");
let lado = 7;
console.log("Lado: " + lado);
let area = lado * lado;
console.log("El área del cuadrado es: " + area);
} else if (figura === 4) {
console.log("Calculando área del CÍRCULO");
let radio = 6;
console.log("Radio: " + radio);
let area = PI * radio * radio;
console.log("El área del círculo es: " + area);
} else {
console.log("Opción no válida. Por favor seleccione 1, 2, 3 o 4.");
}
```

Ejecución del código para hallar el área de figuras en este caso la variable "let figura=1" por lo cual la figura seleccionada es el triángulo.

```
JS index1.js > ...
5
6
7 let figura = 1;
8
9 console.log("Figura seleccionada: " + figura);
10
11 if (figura === 1) {
12     console.log("Calculando área del TRIÁNGULO");
13     let base = 10;
14     let altura = 8;
15     console.log("Base: " + base);
16     console.log("Altura: " + altura);
17     let area = (base * altura) / 2;
18     console.log("El área del triángulo es: " + area);
19 } else if (figura === 2) {
20     console.log("Calculando área del RECTÁNGULO");
21 }
22
23
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(~/Documentos/ADSO/00ADSO/planMejoramiento/AlgorismoEspecialPython/ultimointento)
(12:45:53 on main X *) → node index1.js
=== CALCULADORA DE ÁREAS DE FIGURAS GEOMÉTRICAS ===
Figura seleccionada: 1
Calculando área del TRIÁNGULO
Base: 10
Altura: 8
El área del triángulo es: 40
(~/Documentos/ADSO/00ADSO/planMejoramiento/AlgorismoEspecialPython/ultimointento)
(12:46:17 on main X *) →
```

Edición de la variable "let figura = 2" con lo cual se hallaría el área del rectángulo

```
JS index1.js > figura
2 const PI = 3.14159;
3
4 console.log("=== CALCULADORA DE ÁREAS DE FIGURAS GEOMÉTRICAS ===");
5
6
7 let figura = 1;
8
9 console.log("Figura seleccionada: " + figura);
10
11 if (figura === 1) {
12     console.log("Calculando área del TRIÁNGULO");
13     let base = 10;
14     let altura = 8;
15     console.log("Base: " + base);
16     console.log("Altura: " + altura);
17     let area = (base * altura) / 2;
18     console.log("El área del triángulo es: " + area);
19 }
20
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

GNU nano 8.3 index1.js

```
const PI = 3.14159;
console.log("=== CALCULADORA DE ÁREAS DE FIGURAS GEOMÉTRICAS ===");

let figura = 2;

console.log("Figura seleccionada: " + figura);
```

[50 líneas leídas]

^G Ayuda	^O Guardar	^F Buscar	^K Cortar	^T Ejecutar	^C Ubicación	M-U Deshacer	M-A Poner marca	M-I A llave	M-B Anterior
^X Salir	^R Leer fich.	^N Reemplazar	^U Pegar	^J Justificar	^V Ir a línea	M-E Rehacer	M-G Copiar	^B Buscar atrás	M-F Siguiente

Ejecucion de la variable "let figura = 2"; hhalando el area del rectangulo

```
JS index1.js > ...
2  const PI = 3.14159;
3
4  console.log("=== CALCULADORA DE ÁREAS DE FIGURAS GEOMÉTRICAS ===");
5
6
7  let figura = 2;
8
9  console.log("Figura seleccionada: " + figura);
10
11 if (figura === 1) {
12
13     console.log("Calculando área del TRIÁNGULO");
14     let base = 10;
15     let altura = 8;
16     console.log("Base: " + base);
17     console.log("Altura: " + altura);
18     let area = (base * altura) / 2;
19     console.log("El área del triángulo es: " + area);
20
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
[~/Documentos/ADSO/00ADSO/planMejoramiento/AlgorismoEspecialPython/ultimointento] (mirmonkey@fedora:pts/2)
[12:50:56 on main X *)→ nano index1.js (dom, sep21)
[~/Documentos/ADSO/00ADSO/planMejoramiento/AlgorismoEspecialPython/ultimointento] (mirmonkey@fedora:pts/2)
[12:51:40 on main X *)→ node index1.js (dom, sep21)
=== CALCULADORA DE ÁREAS DE FIGURAS GEOMÉTRICAS ===
Figura seleccionada: 2
Calculando área del RECTÁNGULO
Largo: 12
Ancho: 5
El área del rectángulo es: 60
[~/Documentos/ADSO/00ADSO/planMejoramiento/AlgorismoEspecialPython/ultimointento] (mirmonkey@fedora:pts/2)
[12:51:47 on main X *)→
```

2. Desarrollar un programa que permita almacenar las edades de un grupo de 10 personas en un vector de enteros y luego determine la cantidad de personas que son menores de edad, mayores de edad, cuántos adultos mayores, la edad más baja, la edad más alta y el promedio de edades ingresadas. Para el ejercicio anterior suponga que un adulto mayor debe tener una edad igual o superior a 60. Debe validar para cada ingreso que los valores estén en un rango entre 1 y 120 años. En caso de error deberá notificar y solicitar un nuevo valor.

Desarrollo del codigo para la resolucion del segundo problema planteado:

```
console.log("=== ANÁLISIS ESTADÍSTICO DE EDADES ===");
```

```
let edades = [25, 17, 45, 65, 30, 19, 72, 8, 35, 50];
let MAX_PERSONAS = 10;

let sumaEdades = 0;
let edadMinima = 121;
let edadMaxima = 0;
let menores = 0;
let mayores = 0;
let adultosMayores = 0;

console.log("Edades a analizar:");
let i = 0;
while (i < MAX_PERSONAS) {
  console.log("Persona " + (i + 1) + ": " + edades[i] + " años");
  i = i + 1;
}

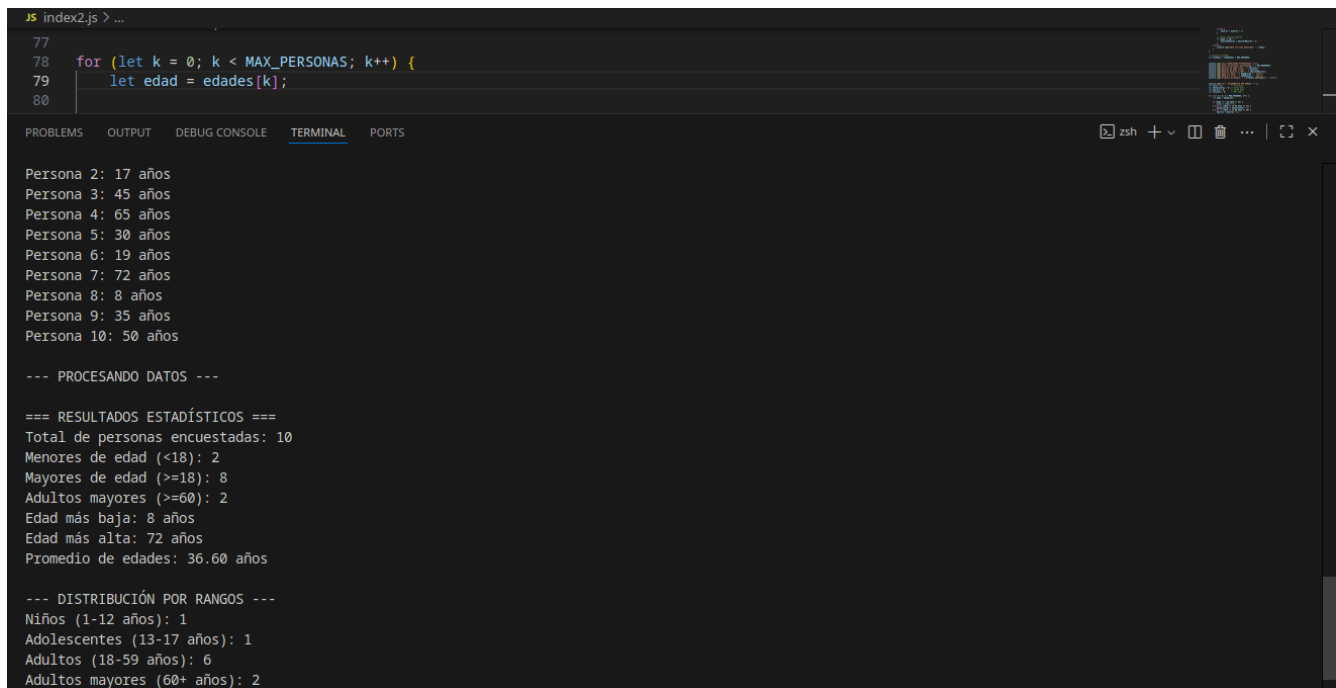
console.log("\n--- PROCESANDO DATOS ---");

for (let j = 0; j < MAX_PERSONAS; j++) {
  let edad = edades[j];

  if (edad >= 1 && edad <= 120) {
    // Sumar para el promedio
    sumaEdades = sumaEdades + edad;
    // Encontrar edad mínima
    if (edad < edadMinima) {
      edadMinima = edad;
    }
    // Encontrar edad máxima
    if (edad > edadMaxima) {
      edadMaxima = edad;
    }
    // Clasificar por grupos de edad
    if (edad < 18) {
      menores = menores + 1;
    } else {
      mayores = mayores + 1;
    }
    // Contar adultos mayores
    if (edad >= 60) {
      adultosMayores = adultosMayores + 1;
    }
    } else {
      console.log("Edad inválida detectada: " + edad);
    }
  }

  // Calcular promedio
  let promedio = sumaEdades / MAX_PERSONAS;
```

Ejecucion del codigo que resuelve el segundo problema planteado mostrando por consola las edades de personas encuestads, clasificacion por rango de edades.



```
JS index2.js > ...
77
78   for (let k = 0; k < MAX_PERSONAS; k++) {
79     let edad = edades[k];
80
Persona 2: 17 años
Persona 3: 45 años
Persona 4: 65 años
Persona 5: 30 años
Persona 6: 19 años
Persona 7: 72 años
Persona 8: 8 años
Persona 9: 35 años
Persona 10: 50 años

--- PROCESANDO DATOS ---

=== RESULTADOS ESTADÍSTICOS ===
Total de personas encuestadas: 10
Menores de edad (<18): 2
Mayores de edad (>=18): 8
Adultos mayores (>=60): 2
Edad más baja: 8 años
Edad más alta: 72 años
Promedio de edades: 36.60 años

--- DISTRIBUCIÓN POR RANGOS ---
Niños (1-12 años): 1
Adolescentes (13-17 años): 1
Adultos (18-59 años): 6
Adultos mayores (60+ años): 2
```

3. Escriba un programa que lea dos vectores de números enteros ordenados ascendentemente y luego produzca la lista ordenada de la mezcla de los dos, por ejemplo, si los dos arreglos tienen los números 1 3 6 9 17 y 2 4 10 17, respectivamente, la lista de números en la pantalla debe ser 1 2 3 4 6 9 10 17 17. Limite los vectores a un tamaño de 5 y debe validar en cada ingreso que realmente se estén ingresando los datos de forma ascendente.

Resolucion del problema planteado mezcla de vectore; se suministra dos array ordenados de manera ascendente los cuales son fusionados por el codigo suministrado


```

console.log("=== MEZCLA DE DOS VECTORES ORDENADOS (SIN
DUPLICADOS) ===");

let TAMANO_VECTOR = 5;
let vectorA = [1, 3, 5, 7, 9];
let vectorB = [1, 4, 6, 8, 10];
let vectorMezclado = [];

console.log("Datos de entrada:");
console.log("Vector A: [1, 3, 5, 7, 9]");
console.log("Vector B: [1, 4, 6, 8, 10]");
console.log("Nota: El número 1 aparece en ambos vectores - se incluirá
solo una vez");

console.log("\n--- VALIDANDO ORDEN ASCENDENTE ---");

let ordenCorrectoA = true;
for (let i = 1; i < TAMANO_VECTOR; i++) {
  if (vectorA[i] < vectorA[i - 1]) {
    ordenCorrectoA = false;
    console.log("ERROR: Vector A no está en orden ascendente");
    break;
  }
}
if (ordenCorrectoA) {
  console.log("Vector A: Orden ascendente correcto ✓");
}

let ordenCorrectoB = true;
for (let j = 1; j < TAMANO_VECTOR; j++) {
  if (vectorB[j] < vectorB[j - 1]) {
    ordenCorrectoB = false;
    console.log("ERROR: Vector B no está en orden ascendente");
    break;
  }
}
if (ordenCorrectoB) {
  console.log("Vector B: Orden ascendente correcto ✓");
}

if (ordenCorrectoA && ordenCorrectoB) {
  console.log("\n--- PROCESO DE MEZCLA SIN DUPLICADOS ---");

  let i = 0; // Índice para Vector A
  let j = 0; // Índice para Vector B
  let k = 0; // Índice para Vector Mezclado

```

```

while (i < TAMANO_VECTOR || j < TAMANO_VECTOR) {
  if (i < TAMANO_VECTOR && j < TAMANO_VECTOR) {

    if (vectorA[i] < vectorB[j]) {

      vectorMezclado[k] = vectorA[i];
      console.log("Paso " + (k + 1) + ": Agregando " + vectorA[i] + " del Vector
A");
      i = i + 1;
      k = k + 1;
    } else if (vectorA[i] > vectorB[j]) {

      vectorMezclado[k] = vectorB[j];
      console.log("Paso " + (k + 1) + ": Agregando " + vectorB[j] + " del Vector
B");
      j = j + 1;
      k = k + 1;
    } else {

      vectorMezclado[k] = vectorA[i];
      console.log("Paso " + (k + 1) + ": Agregando " + vectorA[i] + " (duplicado
encontrado - se incluye solo una vez)");
      i = i + 1;
      j = j + 1;
      k = k + 1;
    }
  } else if (i < TAMANO_VECTOR) {

    vectorMezclado[k] = vectorA[i];
    console.log("Paso " + (k + 1) + ": Agregando " + vectorA[i] + " del Vector A
(restante)");
    i = i + 1;
    k = k + 1;
  } else if (j < TAMANO_VECTOR) {

    vectorMezclado[k] = vectorB[j];
    console.log("Paso " + (k + 1) + ": Agregando " + vectorB[j] + " del Vector B
(restante)");
    j = j + 1;
    k = k + 1;
  }
}
console.log("\n=== RESULTADOS DE LA MEZCLA ===");

```

```
let textoVectorA = "";
for (let x = 0; x < TAMANO_VECTOR; x++) {
  if (x === 0) {
    textoVectorA = vectorA[x];
  } else {
    textoVectorA = textoVectorA + ", " + vectorA[x];
  }
}

console.log("Vector A original: [" + textoVectorA + "]);
```

```
let textoVectorB = "";
for (let y = 0; y < TAMANO_VECTOR; y++) {
  if (y === 0) {
    textoVectorB = vectorB[y];
  } else {
    textoVectorB = textoVectorB + ", " + vectorB[y];
  }
}

console.log("Vector B original: [" + textoVectorB + "]);
```

```
let textoVectorMezclado = "";
for (let z = 0; z < k; z++) {
  if (z === 0) {
    textoVectorMezclado = vectorMezclado[z];
  } else {
    textoVectorMezclado = textoVectorMezclado + ", " + vectorMezclado[z];
  }
}

console.log("Vector Mezclado Final (sin duplicados): [" +
textoVectorMezclado + "]);
console.log("Tamaño del vector resultado: " + k + " elementos");
```

```
console.log("\n--- VERIFICACIÓN FINAL ---");
let mezclaBienOrdenada = true;
for (let w = 1; w < k; w++) {
  if (vectorMezclado[w] < vectorMezclado[w - 1]) {
    mezclaBienOrdenada = false;
    break;
  }
}

if (mezclaBienOrdenada) {
  console.log("✓ La mezcla resultante está correctamente ordenada");
} else {
  console.log("✗ Error: La mezcla no está ordenada correctamente");
}

console.log("\n--- VERIFICACIÓN DE DUPLICADOS ---");
let hayDuplicados = false;
for (let d1 = 0; d1 < k - 1; d1++) {
  for (let d2 = d1 + 1; d2 < k; d2++) {
    if (vectorMezclado[d1] === vectorMezclado[d2]) {
      hayDuplicados = true;
    }
  }
}
```

```
for (let w = 1; w < k; w++) {  
  if (vectorMezclado[w] < vectorMezclado[w - 1]) {  
    mezclaBienOrdenada = false;  
    break;  
  }  
}  
  
if (mezclaBienOrdenada) {  
  console.log("✓ La mezcla resultante está correctamente ordenada");  
} else {  
  console.log("✗ Error: La mezcla no está ordenada correctamente");  
}  
  
console.log("\n--- VERIFICACIÓN DE DUPLICADOS ---");  
let hayDuplicados = false;  
for (let d1 = 0; d1 < k - 1; d1++) {  
  for (let d2 = d1 + 1; d2 < k; d2++) {  
    if (vectorMezclado[d1] === vectorMezclado[d2]) {  
      hayDuplicados = true;  
  
      console.log("✗ Duplicado encontrado: " + vectorMezclado[d1]);  
      break;  
    }  
  }  
}  
  
if (hayDuplicados) {  
  break;  
}  
  
if (!hayDuplicados) {  
  console.log("✓ No se encontraron duplicados en el vector mezclado");  
}  
  
} else {  
  console.log("No se puede realizar la mezcla debido a errores en el orden  
de los vectores.");  
}
```

Ejecucion del codigo correspondiente al problema 3 mezcla de vectores:

```
JS index3.js > ...  
1 console.log("=== MEZCLA DE DOS VECTORES ORDENADOS (SIN DUPLICADOS) ===");  
2  
3 // Vectores predefinidos ordenados ascendentemente con elementos duplicados  
4 let TAMANO_VECTOR = 5;  
  
--- VALIDANDO ORDEN ASCENDENTE ---  
Vector A: Orden ascendente correcto ✓  
Vector B: Orden ascendente correcto ✓  
  
--- PROCESO DE MEZCLA SIN DUPLICADOS ---  
Paso 1: Agregando 1 (duplicado encontrado - se incluye solo una vez)  
Paso 2: Agregando 3 del Vector A  
Paso 3: Agregando 4 del Vector B  
Paso 4: Agregando 5 del Vector A  
Paso 5: Agregando 6 del Vector B  
Paso 6: Agregando 7 del Vector A  
Paso 7: Agregando 8 del Vector B  
Paso 8: Agregando 9 del Vector A  
Paso 9: Agregando 10 del Vector B (restante)  
  
=== RESULTADOS DE LA MEZCLA ===  
Vector A original: [1, 3, 5, 7, 9]  
Vector B original: [1, 4, 6, 8, 10]  
Vector Mezclado Final (sin duplicados): [1, 3, 4, 5, 6, 7, 8, 9, 10]  
Tamaño del vector resultado: 9 elementos  
  
--- VERIFICACIÓN FINAL ---  
✓ La mezcla resultante está correctamente ordenada  
  
--- VERIFICACIÓN DE DUPLICADOS ---  
✓ No se encontraron duplicados en el vector mezclado
```

4. Una emisora con presencia en diferentes ciudades desea conocer el rating de canciones y cantantes más escuchados (sonados) en este semestre del año. Por lo tanto, se ha pedido a estudiantes del SENA del programa de tecnólogo en análisis y desarrollo de software desarrollar una solución que permita conocer la respuesta de 6 personas con relación a sus gustos musicales. Con fines administrativos y realizar una rifa entre las personas encuestadas, la emisora desea poder registrar de las personas entrevistadas su nombre, número de identificación (cédula), fecha de nacimiento, correo electrónico, ciudad de residencia, ciudad de origen. Además, se deberá poder almacenar el artista y título de hasta 3 canciones favoritas en

```
console.log("=== ENCUESTA DE GUSTOS MUSICALES ===");

let MAX_PERSONAS = 6;
let contadorPersonas = 3;

let nombres = ["Juan Pérez", "María González", "Carlos Rodríguez"];
let cedula = ["12345678", "87654321", "11223344"];
let fechasNacimiento = ["15/03/1990", "22/07/1985", "08/12/1992"];
let correos = ["juan@email.com", "maria@email.com", "carlos@email.com"];
let ciudadesResidencia = ["Bogotá", "Medellín", "Cali"];
let ciudadesOrigen = ["Cartagena", "Barranquilla", "Bucaramanga"];

let cancionesArtista1 = ["Shakira", "Manu Chao", "Jesse & Joy"];
let cancionesTitulo1 = ["Hips Don't Lie", "Me Gustas Tu", "Corre"];
let cancionesArtista2 = ["Juanes", "Jesse & Joy", ""];
let cancionesTitulo2 = ["La Camisa Negra", "Te Esperé", ""];
let cancionesArtista3 = ["", "", "Shakira"];
let cancionesTitulo3 = ["", "", "Waka Waka"];

console.log("Sistema con datos de " + contadorPersonas + " personas encuestadas");
console.log("Mostrando información registrada...\n");

console.log("=== INFORMACIÓN DE PERSONAS ENCUESTADAS ===");

for (let i = 0; i < contadorPersonas; i++) {
  console.log("\n--- PERSONA " + (i + 1) + " ---");
  console.log("Nombre: " + nombres[i]);
  console.log("Cédula: " + cedula[i]);
  console.log("Fecha de Nacimiento: " + fechasNacimiento[i]);
  console.log("Correo: " + correos[i]);
  console.log("Ciudad de Residencia: " + ciudadesResidencia[i]);
  console.log("Ciudad de Origen: " + ciudadesOrigen[i]);
}
```

```

console.log("Canciones Favoritas:");
let contadorCanciones = 0;
if (cancionesTitulo1[i] !== "") {
console.log(" 1. " + cancionesTitulo1[i] + " - " + cancionesArtista1[i]);
contadorCanciones = contadorCanciones + 1;
}
if (cancionesTitulo2[i] !== "") {
console.log(" 2. " + cancionesTitulo2[i] + " - " + cancionesArtista2[i]);
contadorCanciones = contadorCanciones + 1;
}
if (cancionesTitulo3[i] !== "") {
console.log(" 3. " + cancionesTitulo3[i] + " - " + cancionesArtista3[i]);
contadorCanciones = contadorCanciones + 1;
}
if (contadorCanciones === 0) {
console.log(" No registró canciones favoritas");
}
}

console.log("\n=== ESTADÍSTICAS DE LA ENCUESTA ===");

let personasConCanciones = 0;
let totalCanciones = 0;

for (let i = 0; i < contadorPersonas; i++) {
let cancionesPersona = 0;
if (cancionesTitulo1[i] !== "") {
cancionesPersona = cancionesPersona + 1;
}
if (cancionesTitulo2[i] !== "") {
cancionesPersona = cancionesPersona + 1;
}
}

```

```
if (cancionesTitulo3[i] !== "") {  
  cancionesPersona = cancionesPersona + 1;  
}  
if (cancionesPersona > 0) {  
  personasConCanciones = personasConCanciones + 1;  
}  
totalCanciones = totalCanciones + cancionesPersona;  
}  
  
console.log("Total de personas encuestadas: " + contadorPersonas);  
console.log("Personas que registraron canciones: " + personasConCanciones);  
console.log("Total de canciones registradas: " + totalCanciones);  
  
let promedioCanciones = totalCanciones / contadorPersonas;  
console.log("Promedio de canciones por persona: " +  
  promedioCanciones.toFixed(2));  
  
console.log("\n--- ANÁLISIS DE ARTISTAS ---");  
  
let artistasUnicos = [];  
let conteoArtistas = [];  
  
for (let i = 0; i < contadorPersonas; i++) {  
  if (cancionesArtista1[i] !== "") {  
    let encontrado = false;  
    for (let j = 0; j < artistasUnicos.length; j++) {  
      if (artistasUnicos[j] === cancionesArtista1[i]) {  
        conteoArtistas[j] = conteoArtistas[j] + 1;  
        encontrado = true;  
        break;  
      }  
    }  
  }  
}
```



```
if (!encontrado) {  
  artistasUnicos[artistasUnicos.length] = cancionesArtista1[i];  
  conteoArtistas[conteoArtistas.length] = 1;  
}  
}  
if (cancionesArtista2[i] !== "") {  
  let encontrado = false;  
  for (let j = 0; j < artistasUnicos.length; j++) {  
    if (artistasUnicos[j] === cancionesArtista2[i]) {  
      conteoArtistas[j] = conteoArtistas[j] + 1;  
      encontrado = true;  
      break;  
    }  
  }  
}  
if (!encontrado) {  
  artistasUnicos[artistasUnicos.length] = cancionesArtista2[i];  
  conteoArtistas[conteoArtistas.length] = 1;  
}  
}  
if (cancionesArtista3[i] !== "") {  
  let encontrado = false;  
  for (let j = 0; j < artistasUnicos.length; j++) {  
    if (artistasUnicos[j] === cancionesArtista3[i]) {  
      conteoArtistas[j] = conteoArtistas[j] + 1;  
      encontrado = true;  
      break;  
    }  
  }  
}  
if (!encontrado) {  
  artistasUnicos[artistasUnicos.length] = cancionesArtista3[i];  
  conteoArtistas[conteoArtistas.length] = 1;  
}  
}  
}
```

```

console.log("Artistas mencionados:");
for (let k = 0; k < artistasUnicos.length; k++) {
console.log("- " + artistasUnicos[k] + ": " + conteoArtistas[k] + " mención(es)");
}

if (artistasUnicos.length > 0) {
let maxMenciones = 0;
let artistaPopular = "";
for (let m = 0; m < conteoArtistas.length; m++) {
if (conteoArtistas[m] > maxMenciones) {
maxMenciones = conteoArtistas[m];
artistaPopular = artistasUnicos[m];
}
}
console.log("\n🏆 Artista más mencionado: " + artistaPopular + " (" + maxMenciones + " menciones)");
}

console.log("\n--- FIN DE LA ENCUESTA ---");

```

Ejecucion del codigo propuesto para la solucion al problema #4 encuesta musical

```

JS index4.js > ...
142     artistasUnicos[artistasUnicos.length] = cancionesArtista3[i];
143     conteoArtistas[conteoArtistas.length] = 1;
144
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
[~/Documentos/ADSO/00ADSO/planMejoramiento/AlgoritmoEspecialPython/ultimointento] (mxmonkey@fedora:pts/3)
(17:07:55 on main X *) → node index4.js
Cédula: 11223344
Fecha de Nacimiento: 08/12/1992
Correo: carlos@email.com
Ciudad de Residencia: Cali
Ciudad de Origen: Bucaramanga
Canciones Favoritas:
  1. Corre - Jesse & Joy
  3. Waka Waka - Shakira

=== ESTADÍSTICAS DE LA ENCUESTA ===
Total de personas encuestadas: 3
Personas que registraron canciones: 3
Total de canciones registradas: 6
Promedio de canciones por persona: 2.00

--- ANÁLISIS DE ARTISTAS ---
Artistas mencionados:
- Shakira: 2 mención(es)
- Juanes: 1 mención(es)
- Manu Chao: 1 mención(es)
- Jesse & Joy: 2 mención(es)

🏆 Artista más mencionado: Shakira (2 menciones)

--- FIN DE LA ENCUESTA ---

```

CONCLUSIÓN

La resolución de esta evidencia confirma que se han proporcionado soluciones coherentes y funcionales para cada uno de los problemas algorítmicos planteados, cumpliendo satisfactoriamente con los objetivos técnicos establecidos. Se ha logrado el uso exitoso del lenguaje JavaScript para la implementación de algoritmos complejos, integrando eficientemente las estructuras de control básicas (condicionales y bucles) con estructuras de almacenamiento vectoriales, demostrando un dominio sólido de los fundamentos de la programación.

La evidencia ha puesto de manifiesto la capacidad avanzada de manipulación de arreglos en diferentes dimensiones para dar solución a problemas reales de diversa complejidad. Se ha demostrado proficiencia en el manejo de arrays simples para cálculos geométricos, arrays paralelos para análisis estadístico de datos, algoritmos de mezcla para vectores ordenados, y estructuras de almacenamiento multidimensionales para la gestión de información personal compleja.

Las habilidades aplicadas en lógica de programación incluyen la implementación exitosa de validaciones de entrada de datos dentro de rangos específicos (como edades entre 1 y 120 años), la ejecución de cálculos estadísticos complejos (promedios, valores máximos y mínimos, contadores categóricos), y la organización eficiente de datos estructurados que manejan múltiples atributos por entidad (información personal y preferencias musicales) dentro de estructuras vectoriales paralelas.

El desarrollo de esta evidencia afianza las bases algorítmicas requeridas para la construcción de software, constituyendo la esencia fundamental de la programación moderna. La implementación rigurosa de algoritmos sin dependencia de funciones avanzadas demuestra un entendimiento profundo de los principios computacionales básicos, estableciendo cimientos sólidos para el desarrollo de aplicaciones más complejas.

El producto final materializa la convergencia de conocimientos teóricos y prácticos, cumpliendo con la exigencia académica de un trabajo escrito bien estructurado que documenta no solo las soluciones algorítmicas, sino también el proceso metodológico empleado en su desarrollo. Esta evidencia representa la demostración tangible de las competencias adquiridas en el manejo de estructuras de almacenamiento y la resolución algorítmica de problemas computacionales.

