Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

**Alumna**

Karla Lilia Córdova Fernández

**Unidad de Aprendizaje:** Programación Avanzada

**Profesor**

M. en C. Niels Henrik Navarrete Manzanilla

**Práctica**

Diagramas de Flujo

**Ciudad de México; a 22 de octubre de 2020**

|  |
| --- |
| **ÍNDICE** |

Contenido

[INTRODUCCIÓN 4](#_Toc54204600)

[DESARROLLO 5](#_Toc54204601)

[Programa 1 5](#_Toc54204602)

[Descripción 5](#_Toc54204603)

[Análisis (algoritmo) 5](#_Toc54204604)

[Pseudocódigo 5](#_Toc54204605)

[Diagrama de flujo 7](#_Toc54204606)

[Programa 2 8](#_Toc54204607)

[Descripción 8](#_Toc54204608)

[Análisis (algoritmo) 8](#_Toc54204609)

[Pseudocódigo 8](#_Toc54204610)

[Diagrama de flujo 9](#_Toc54204611)

[Programa 3 10](#_Toc54204612)

[Descripción 10](#_Toc54204613)

[Análisis (algoritmo) 10](#_Toc54204614)

[Pseudocódigo 10](#_Toc54204615)

[Diagrama de flujo 11](#_Toc54204616)

[Programa 4 12](#_Toc54204617)

[Descripción 12](#_Toc54204618)

[Análisis (algoritmo) 12](#_Toc54204619)

[Pseudocódigo 12](#_Toc54204620)

[Diagrama de flujo 13](#_Toc54204621)

[Programa 5 14](#_Toc54204622)

[Descripción 14](#_Toc54204623)

[Análisis (algoritmo) 14](#_Toc54204624)

[Pseudocódigo 14](#_Toc54204625)

[Diagrama de flujo 15](#_Toc54204626)

[Programa 6 16](#_Toc54204627)

[Descripción 16](#_Toc54204628)

[Análisis (algoritmo) 16](#_Toc54204629)

[Pseudocódigo 16](#_Toc54204630)

[Diagrama de flujo 18](#_Toc54204631)

[CONCLUSIONES 19](#_Toc54204632)

|  |
| --- |
| INTRODUCCIÓN |

Para comprender la lógica de un programa, antes de la creación del código es importante la elaboración del algoritmo, donde se plantean las tres preguntas: “¿Cuáles son las entradas y salidas? ¿Qué es lo que hará el programa? ¿Qué espero de salida?” para empezar teniendo una idea base de qué vamos nosotros a introducir en un principio y qué será lo que el programa hará para devolvernos nuestra salida.

El pseudocódigo es el diseño de lo que será el código final y el diagrama de flujo de datos, es el gráfico que muestra la arquitectura y funcionamiento del programa.

Para crear de modo más fácil y ordenado un programa, el uso de estas tres herramientas es lo más adecuado.

|  |
| --- |
| DESARROLLO |

# Programa 1

### Descripción

Calcular la paga neta de un trabajador conociendo el número de horas trabajadas, la tarifa horaria y la tasa de impuestos.

### Análisis (algoritmo)

**1. ¿Cuáles son la entradas y salidas?**

Entradas:

int horas\_trabajadas;

float tarifa\_hora;

float tasa\_impuesto;

Salidas:

float paga\_trabajador;

**2. ¿Qué es lo que hará el programa?**

Multiplicación -> paga\_bruta = horas\_trabajadas \* tarifa\_hora;

Multiplicación -> impuesto = paga\_bruta \* tasa\_impuesto;

Resta -> paga\_trabajador = paga\_bruta – impuesto;

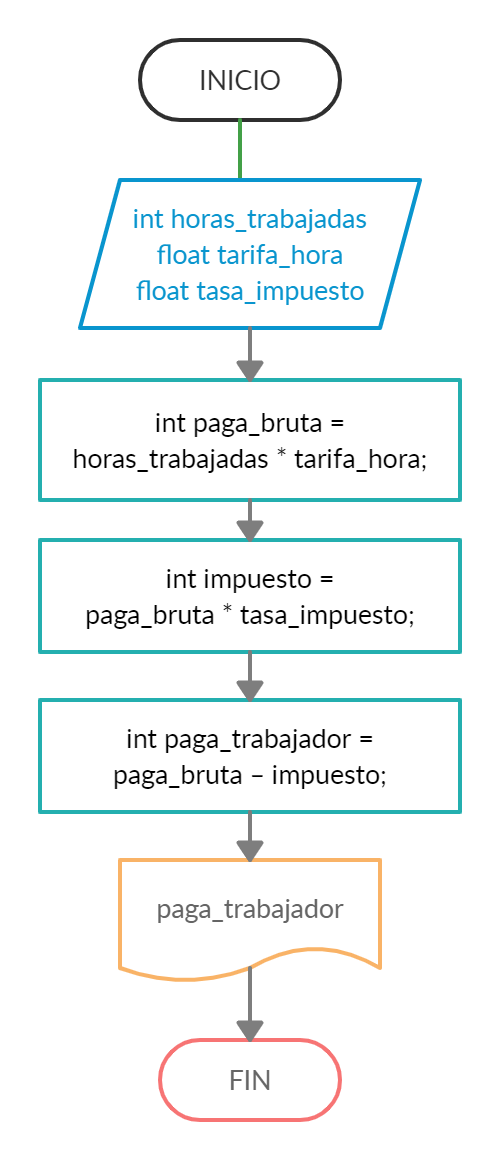
**3. ¿Qué espero de salida?**

Imprimir -> paga\_trabajador

### Pseudocódigo

|  |
| --- |
| int main(){  int horas\_trabajadas;  float tarifa\_hora;  float tasa\_impuesto;  // Obtener valores de entrada desde teclado  scanf(“%f”,&horas);  scanf(“%f”,&tarifa\_hora);  scanf(“%f”,&tasa\_impuesto);  float paga\_bruta = horas\_trabajadas \* tarifa\_hora;  float impuesto = paga\_bruta \* tasa\_impuesto;  float paga\_trabajador = paga\_bruta – impuesto;  //Imprimir dato  printf(“%f”, paga\_trabajador);  return 0;  } |

### Diagrama de flujo



# Programa 2

### Descripción

Calcular el valor de la suma 1+2+3+.......+100.

### Análisis (algoritmo)

**1. ¿Cuáles son mis entradas y salidas?**

Entradas:

int numero = 100;

Salidas:

int suma;

**2. ¿Qué es lo que hará el programa?**

Sumar a una variable valores desde 1, que aumenten de uno en uno hasta 100.

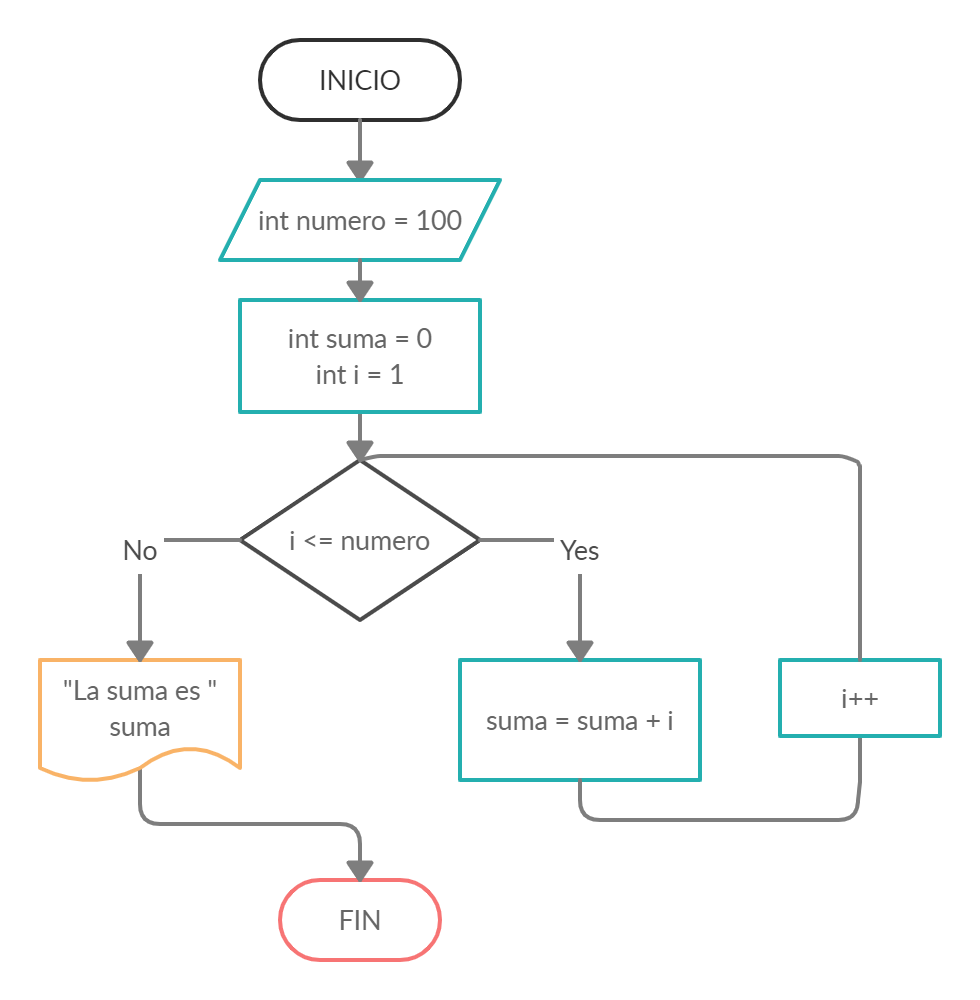
**3. ¿Qué espero de salida?**

Imprimir -> suma

### Pseudocódigo

|  |
| --- |
| int main(){  int numero = 100;  int suma = 0;  // Ciclo para que sume de uno en uno desde 1 hasta 100  for(int i = 1; i<=100; i++){  suma = suma+i;  }  //Imprimir dato  printf(“%d”, suma);  return 0;  } |

### Diagrama de flujo



# Programa 3

### Descripción

Determinar si un número es par o impar.

### Análisis (algoritmo)

**1. ¿Cuáles son mis entradas y salidas?**

Entradas:

int numero;

Salidas:

Mensaje -> “Es par” o “Es impar”.

**2. ¿Qué es lo que hará el programa?**

Comparación -> si (numero % 2 == 0), es decir, si el módulo de la división del número entre dos es cero, es par

De ser cualquier otro residuo, es impar.

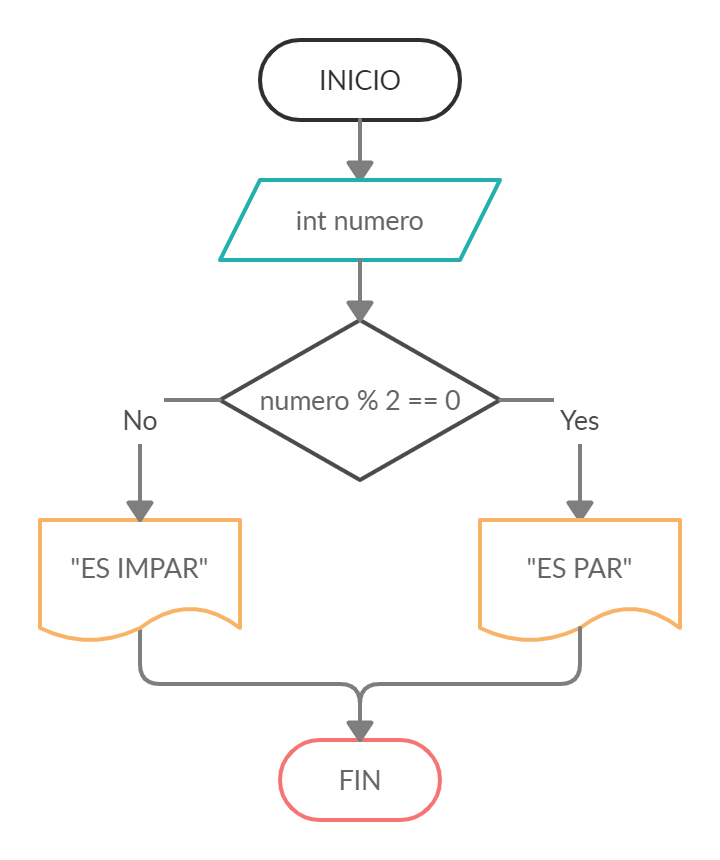
**3. ¿Qué espero de salida?**

Imprimir -> “Es par” o “Es impar” según sea el caso.

### Pseudocódigo

|  |
| --- |
| int main(){  int numero;  //Petición del número  scanf(“%d”,&numero);  // Condición número módulo 2  if(numero % 2 == 0)  //Es par  else  //Es impar  return 0;  } |

### Diagrama de flujo



# Programa 4

### Descripción

Realizar la suma de todos los números pares entre 2 y 100.

### Análisis (algoritmo)

**1. ¿Cuáles son mis entradas y salidas?**

Entradas:

int numero = 100;

Salidas:

int suma;

**2. ¿Qué es lo que hará el programa?**

Un ciclo recorrerá todos los números del 2 al 100.

Comparación -> si ( i (número entre 2 y 100) % 2 == 0), es decir, si el módulo de la división del número entre dos es cero, es par, por lo tanto, se suma.

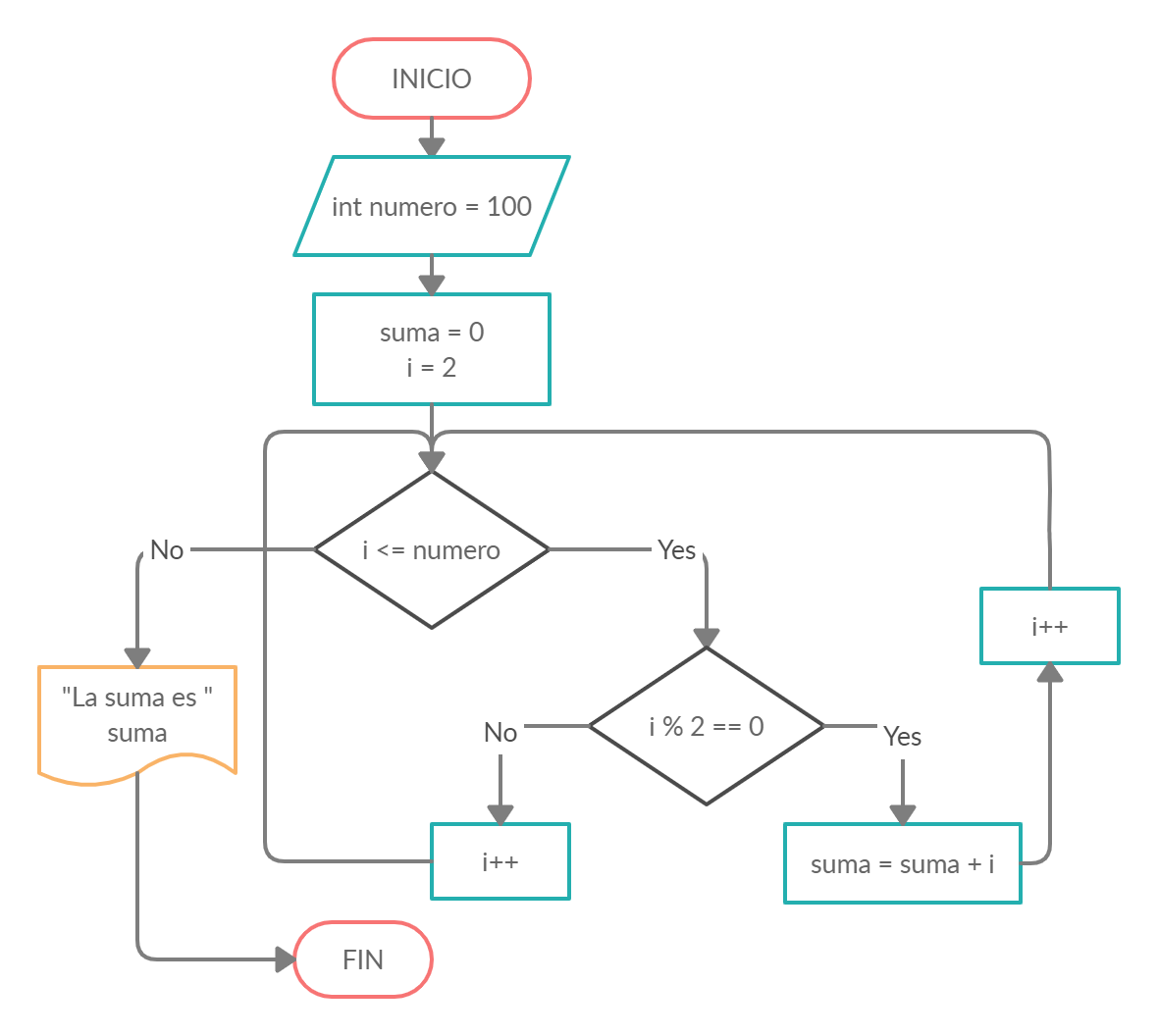
**3. ¿Qué espero de salida?**

Imprimir-> suma.

### Pseudocódigo

|  |
| --- |
| int main(){  int numero = 100;  int suma = 0;  // Ciclo que recorre todos los números del 2 hasta 100  for(int i = 2; i<=numero; i++){  // Condición para saber si un número es par y sumarlo  if(i % 2 == 0)  suma = suma+i;  }  //Imprimir dato  printf(“%d”, suma);  return 0;  } |

### Diagrama de flujo



# Programa 5

### Descripción

Calcular el factorial de un número dado.

### Análisis (algoritmo)

**1. ¿Cuáles son mis entradas y salidas?**

Entradas:

int numero;

Salidas:

int factorial;

**2. ¿Qué es lo que hará el programa?**

Comparación -> (numero == 0) factorial es uno.

Si (numero > 0) se crea un ciclo, donde numero- - por cada ciclo mientras sea > 1.

En cada ciclo:

Multiplicación -> factorial = factorial\*numero

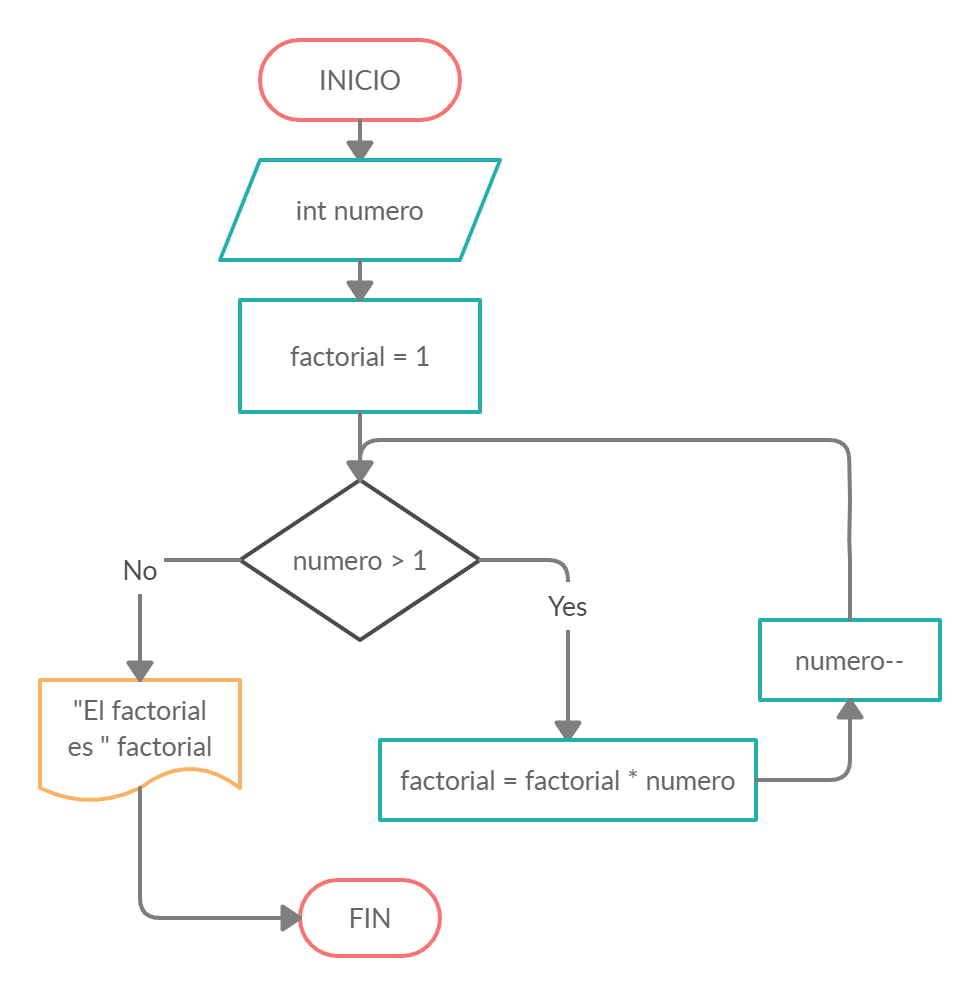
**3. ¿Qué espero de salida?**

int factorial;

### Pseudocódigo

|  |
| --- |
| int main(){  int factorial = 1;  int numero = 0;  // Se pide desde teclado el entero  scanf("%i",&numero);  else{  // Condición: por cada ciclo se le resta una unidad a número, mientras número > 1 dura el ciclo  while(numero > 1){  factorial = factorial\*numero;  numero--;  }    }  //Imprimir dato  printf("%i", factorial);  return 0;  } |

### Diagrama de flujo



# Programa 6

### Descripción

Calcular la media de una serie de números positivos, suponiendo que los datos se leen desde una terminal. El valor (cero) -como entrada- indicará el final de la serie de números. Ej. 1 2 3 4 5 0

### Análisis (algoritmo)

**1. ¿Cuáles son mis entradas y salidas?**

Entradas:

int n;

int numeros[n];

Salidas:

int media;

**2. ¿Qué es lo que hará el programa?**

Con un ciclo se recorrerán todas las posiciones del arreglo de enteros numeros[n] hasta encontrar un número cero.

Suma -> Conforme el ciclo vaya recorriendo el arreglo, los datos de cada posición se irán sumando.

División -> La suma total se dividirá entre el número de elementos.

Condición en caso de la excepción de que no exista ningún dato.

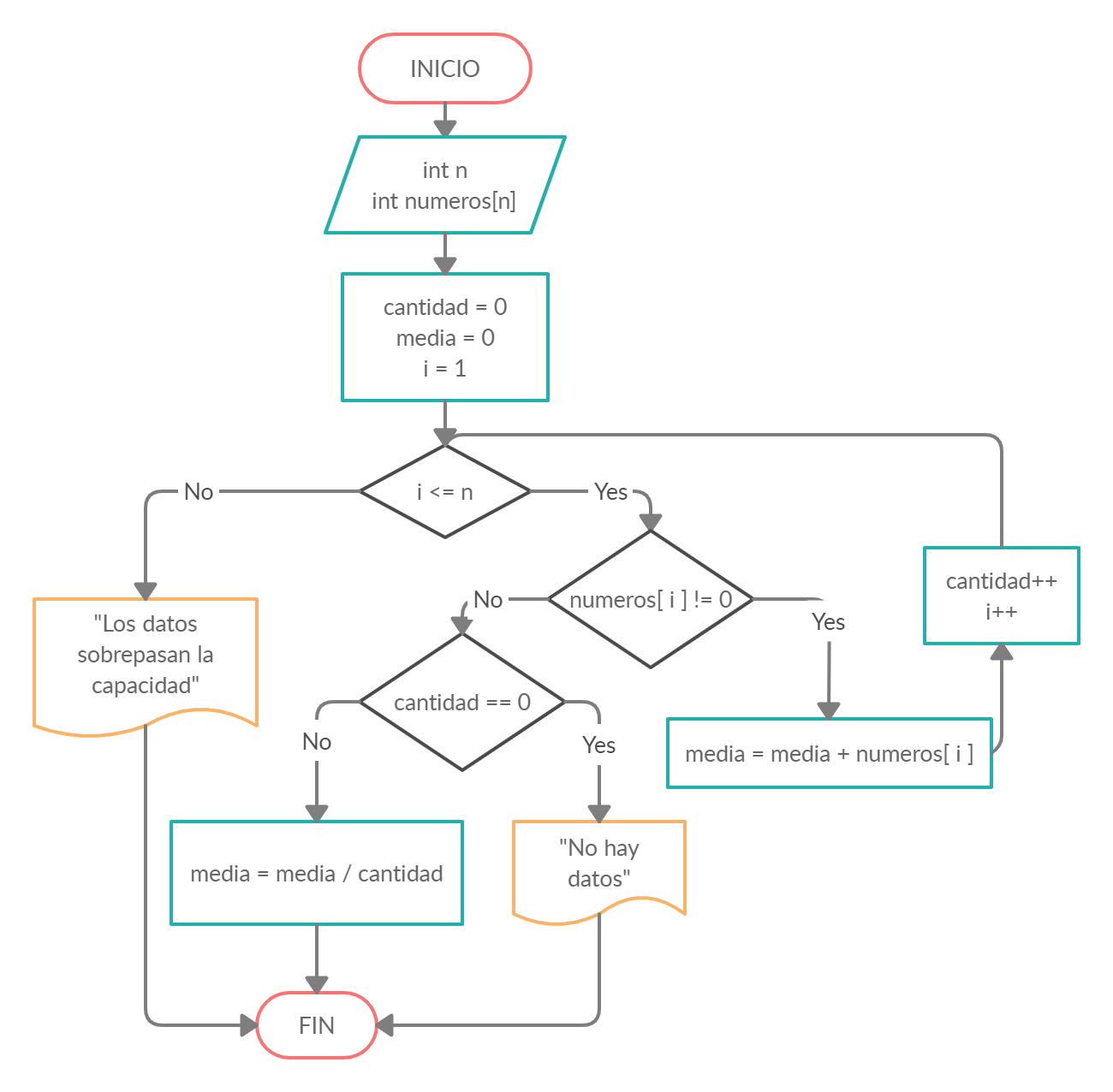
**3. ¿Qué espero de salida?**

int media;

### Pseudocódigo

|  |
| --- |
| int main(){  int n;  int numeros[n];  int cantidad = 0;  float media = 0;  // Ciclo que recorre las posiciones del arreglo  for(int i=0; i<n; i++){  //Condición que evalúa si es cero (el dato final)  if(numeros[i] != 0){  media = media + numeros[i];  cantidad++;}  else  break;  }  //Condición por si no existe ningún dato más que cero.  if(cantidad == 0)  //No hay datos  else  media = media/cantidad;  //Imprimir dato  printf(“%f”, media);  return 0;  } |

### Diagrama de flujo



|  |
| --- |
| CONCLUSIONES |

La aplicación de el algoritmo y pseudocódigo ayudan a mejorar la comprensión del programa que se está haciendo, evita hacernos divagar en las ideas, además de localizar más rápido los errores.

El diagrama de flujo igualmente ayuda a detectar errores y a comprender qué es lo que estamos haciendo con nuestras variables y funciones. El diagrama de flujo es muy funcional para cualquier persona que desconozca el código y quiera entender su funcionamiento.