SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

Instituto Politécnico Nacional
"La Técnica al Servicio de la Patria"

upiita-ipn

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

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?

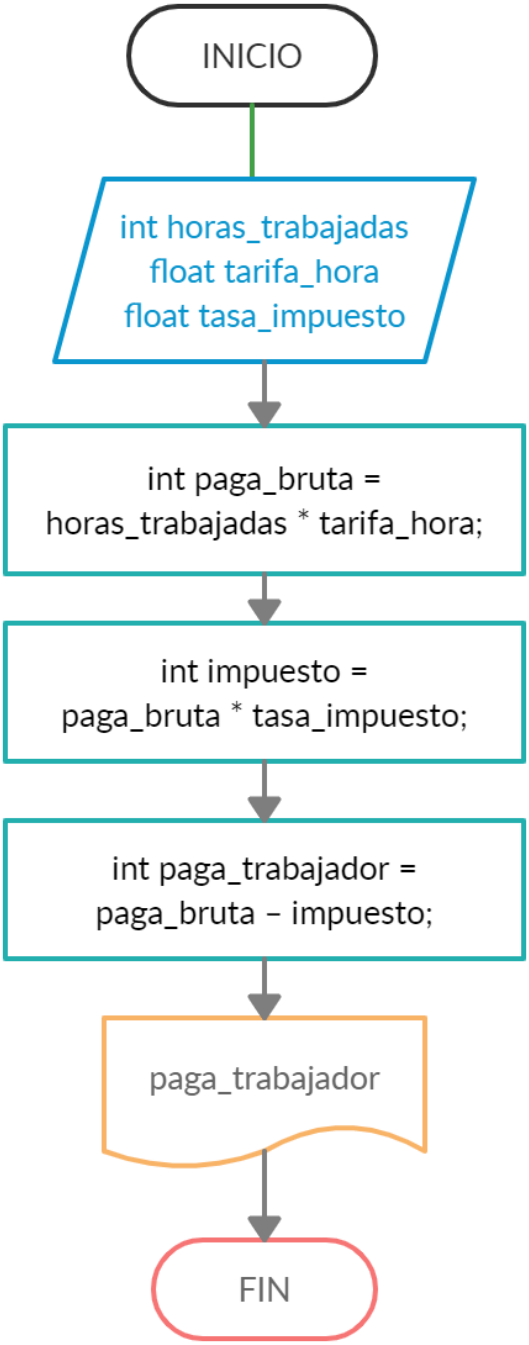
Imprimir -> paga_trabajador

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.

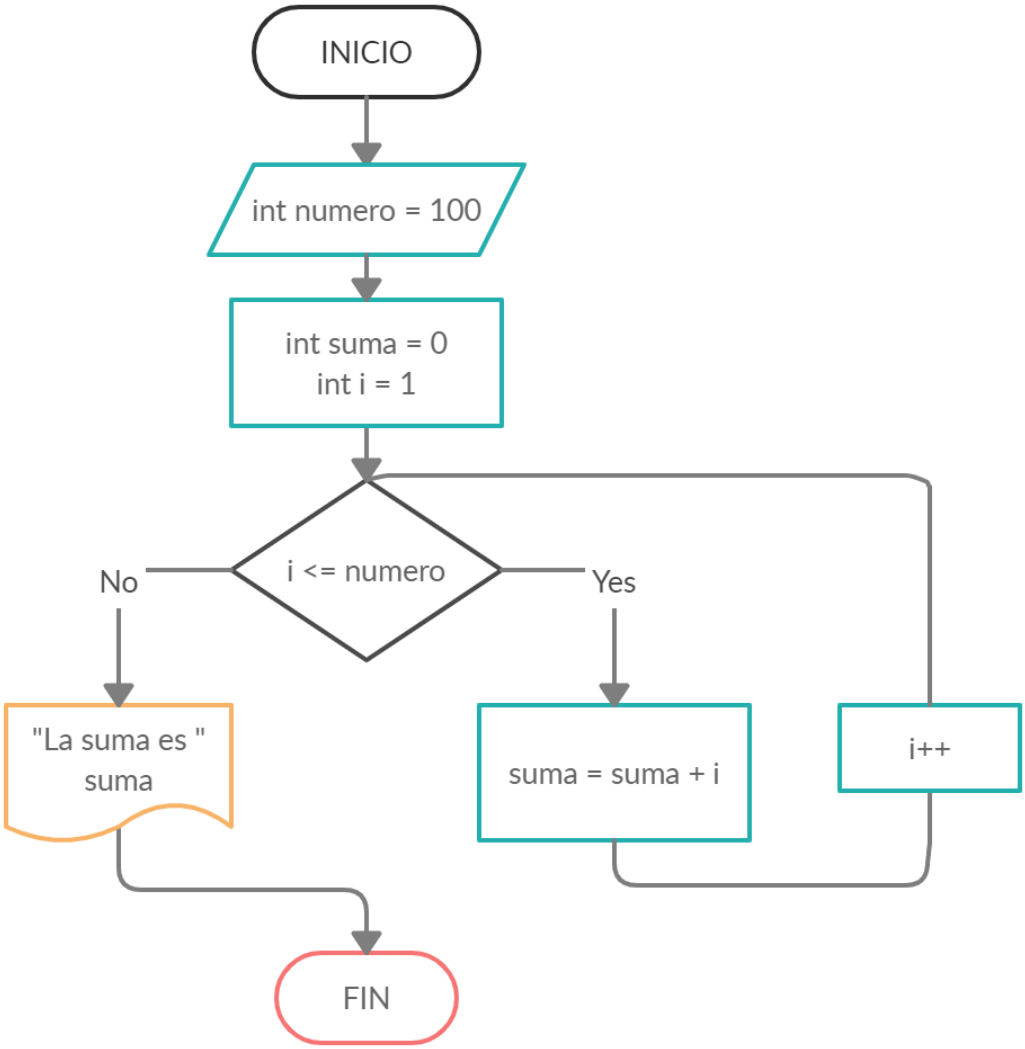
3. ¿Qué espero de salida?

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.

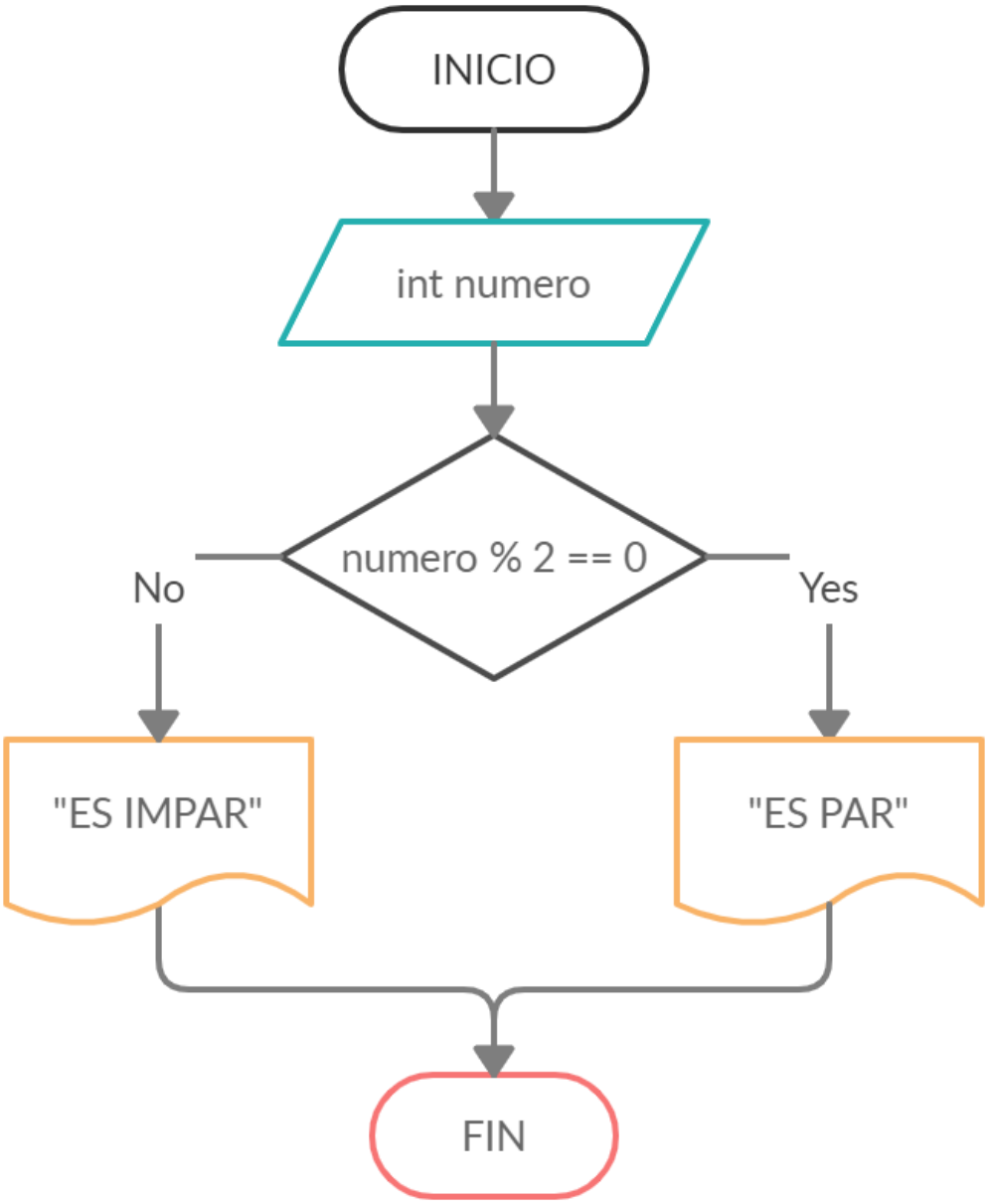
3. ¿Qué espero de salida?

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.

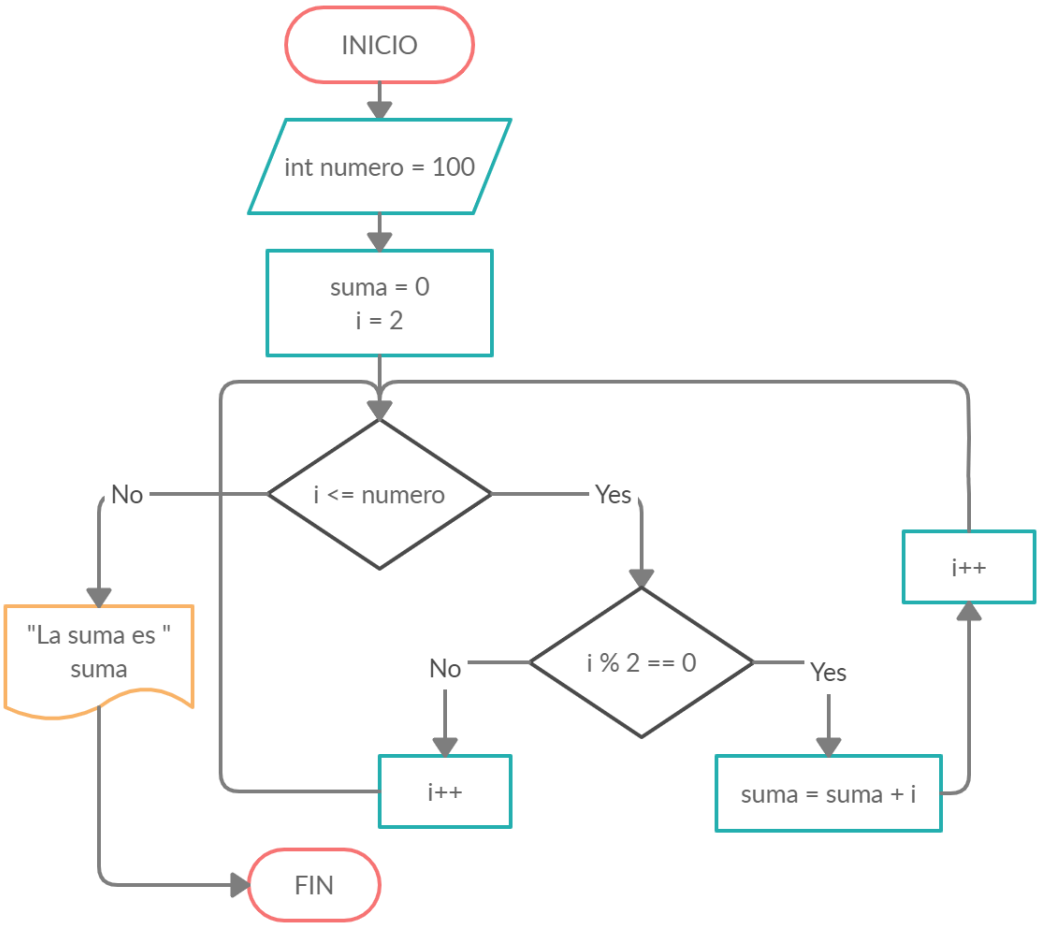
3. ¿Qué espero de salida?

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?

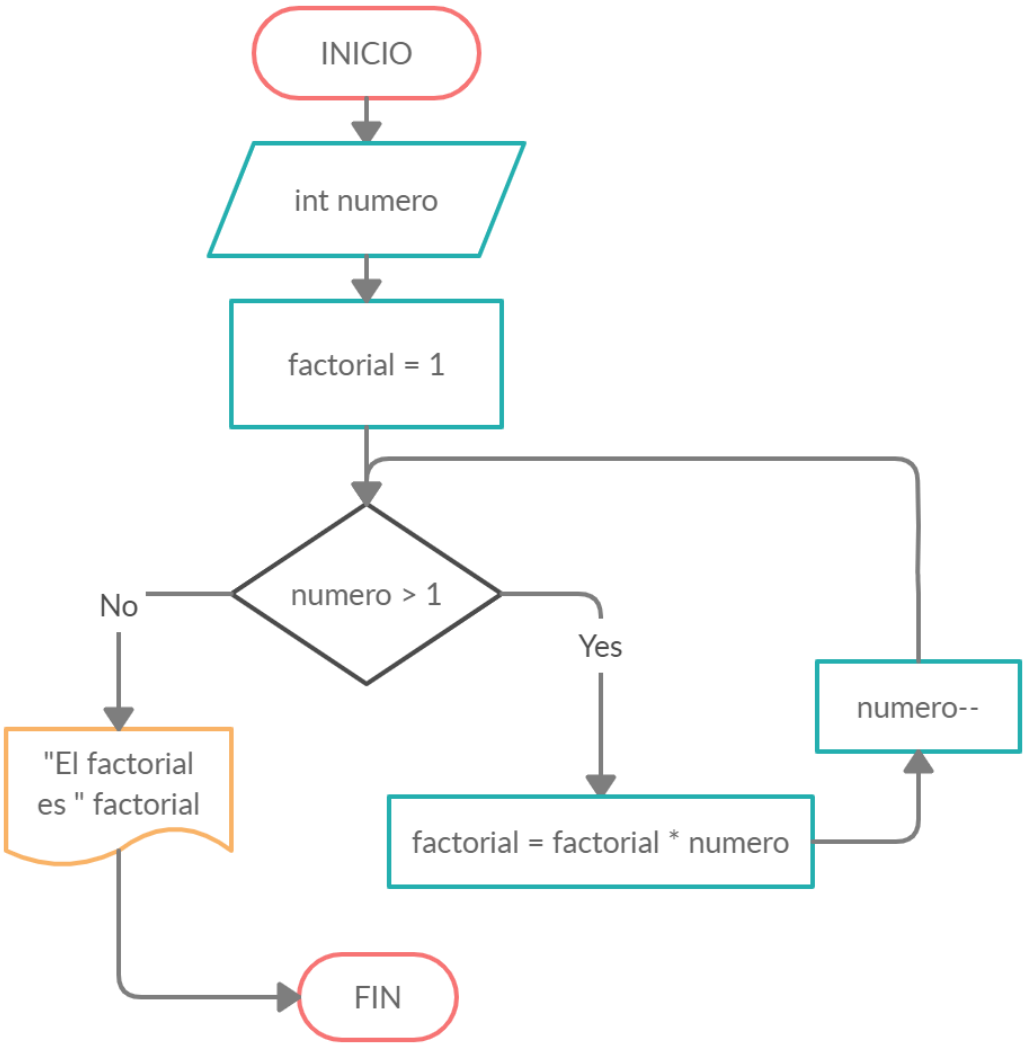
int factorial;

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?

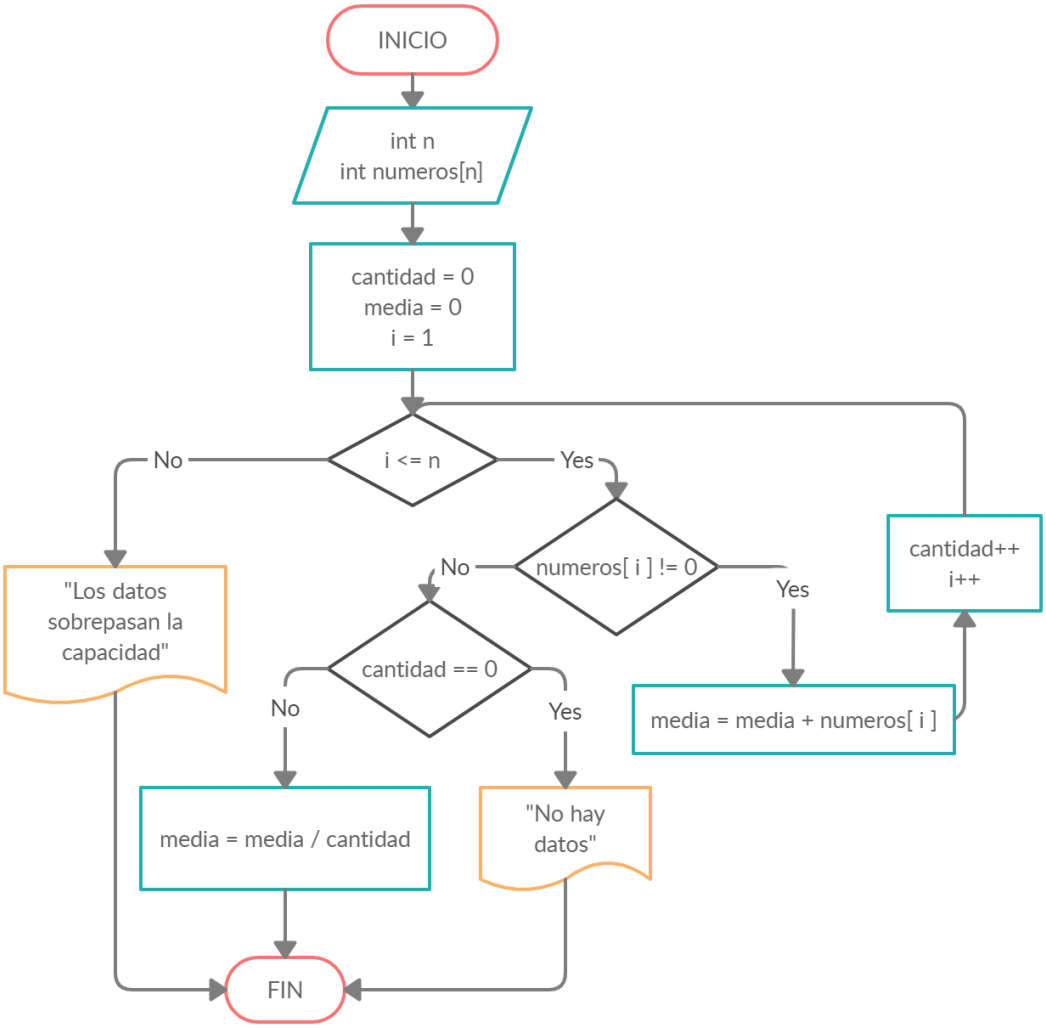
int media;

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.