	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Reporte de práctica 5:Pseudocódigo.

Profesor(a): Oscar René Valdez Casillas

Asignatura: Fundamentos de Programación

Grupo: 21

No de Práctica(s): 5

Integrante(s): ***Santiago Durán Rendón***

Santiago Noriega Chiu

Jesús Ramírez Reyes

No. de lista o brigada: 01

Semestre: 2025-1

Fecha de entrega: 17/09/2024

Observaciones:

CALIFICACIÓN: _____

Brigada 1. (2024). *Reporte de Práctica 5.Pseudocódigo*. UNAM

Índice

Resumen

Este reporte aborda la resolución de problemas a través de pseudocódigos en el ámbito de la programación. La práctica incluye cuatro ejercicios fundamentales: la secuencia de Fibonacci, la venta de boletos en un estadio, la identificación de números primos menores a un valor dado, y la sumatoria desde n=1 hasta N. Se utilizaron algoritmos y diagramas de flujo para estructurar las soluciones, verificando su validez mediante pruebas de escritorio. El objetivo principal fue la implementación de pseudocódigos que resuelvan problemas algorítmicos de manera eficiente y comprensible.

Introducción

El reporte documenta el desarrollo de la quinta práctica de la asignatura de Fundamentos de Programación, centrada en la elaboración de pseudocódigos para la resolución de problemas algorítmicos. El pseudocódigo es una herramienta esencial en la programación, ya que permite expresar soluciones en un formato comprensible tanto para humanos como para máquinas, sin depender de un lenguaje de programación específico. La práctica se compone de cuatro ejercicios: el cálculo del décimo término de la secuencia de Fibonacci, la simulación de la venta de boletos en un estadio, la detección de números primos menores a un número dado, y la sumatoria desde n=1 hasta un valor N. Se busca fortalecer el pensamiento lógico y la capacidad de traducir una solución conceptual a una estructura algorítmica.

Objetivo

Elaborar pseudocódigos que representen soluciones algorítmicas empleando la sintaxis y semántica adecuadas.

Desarrollo de Contenidos:

1. Realizar los pseudocódigos de los ejercicios de la práctica 3. Incluir una prueba de escritorio para validar que el algoritmo funciona de manera correcta.

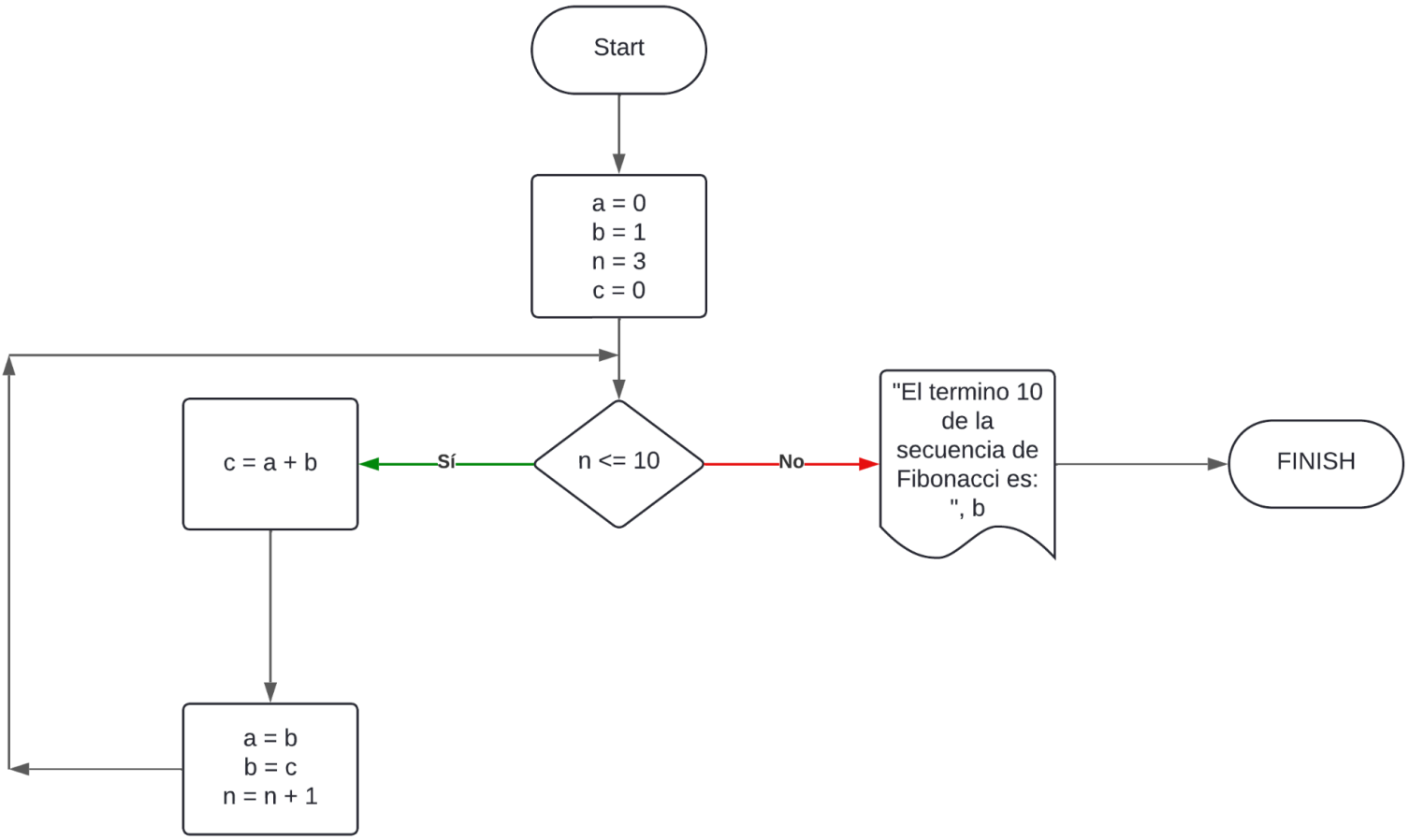
Ejercicio 1. Secuencia de Fibonacci.

Encuentre el algoritmo para calcular el término 10 de la secuencia de Fibonacci. Recuerde que los dos primeros números de la serie son 0 y 1. El resto se calcula con la suma de los dos números inmediatos que le preceden.

Algoritmo:

1. Inicializar las variables enteras: a = 0, b = 1, c = 0
2. Do While 0 < n <= 10
- c = a + b
- a = b
- b = c
- n = n + 1
- imprimir el valor de “b”
3. Cuando n no sea igual a la condición dada imprimir mensaje de error.

Diagrama de flujo:



Pseudocódigo:

```
INICIO Algoritmo_secuencia_de_FIbonacci
    a = 0
    b = 1
    n = 3
    c = 0
    SI n <= 10
        c = a + b
        a = b
        b = c
        n = n + 1
    SI NO
        ESCRIBIR “El término 10 de la sucesión de Fibonacci es: “, c
    FIN SI
FIN
```

Tabla 1: Prueba de escritorio secuencia de Fibonacci.

n	a	b	c
0	0	1	1
1	1	1	2
2	1	2	3
3	2	3	5
4	3	5	8
5	5	8	13
6	8	13	21
7	13	21	34
8	21	34	55
9	34	55	89
10	55	89	144

Ejercicio 2. Venta de Boletos en un Estadio.

En un estadio se tienen 5 tipos diferentes de localidades, las cuales se identifican por una clave numérica que es un valor comprendido entre 1 y 5. Los precios de cada localidad y los datos referentes a las ventas de boletos para el próximo juego son:

Clave 1 → Boleto 1	P1 → Precio boleto 1
Clave 2 → Boleto 2	P2 → Precio boleto2
Clave 3 → Boleto 3	P3 → Precio boleto 3
Clave 4 → Boleto 4	P4 → Precio boleto 4
Clave 5 → Boleto 5	P5 → Precio boleto 5

Algoritmo:

1. Leer los precios de los boletos P1, P2, P3, P4, P5.
2. Inicializar contadores para la cantidad de boletos vendidos de cada tipo.
3. Para cada venta:
    - Leer la clave del boleto y la cantidad vendida.
    - Calcular el importe de la venta: importe = precio * cantidad.
    - Incrementar el contador correspondiente para la cantidad de boletos vendidos.
    - Imprimir la clave, cantidad e importe de la venta.
4. Calcular e imprimir la recaudación total del estadio sumando los importes de todas las ventas.
5. Imprimir la cantidad total de boletos vendidos por cada tipo.

Algoritmo parte dos:

1. Inicialización de las variables:

    P1, P2, P3, P4, P5 // Precios de los boletos

    Cantidad1 = 0 // Cantidad de boletos vendidos para la Clave 1

    Cantidad2 = 0 // Cantidad de boletos vendidos para la Clave 2

    Cantidad3 = 0 // Cantidad de boletos vendidos para la Clave 3

    Cantidad4 = 0 // Cantidad de boletos vendidos para la Clave 4

    Cantidad5 = 0 // Cantidad de boletos vendidos para la Clave 5

    RecaudacionTotal = 0 // Recaudación total del estadio

2. Leer P1, P2, P3, P4, P5
3. Inicializar la venta: Sí Venta=Sí:

    Leer la clave del tipo de boleto vendido: Leer Clave

    Leer la cantidad de boletos vendidos: Leer CantidadVendida

    Calcular el importe de la venta:

    Si Clave = 1 entonces Importe = P1 * CantidadVendida Cantidad1 = Cantidad1 + CantidadVendida

    Si Clave = 2 entonces Importe = P2 * CantidadVendida Cantidad2 = Cantidad2 + CantidadVendida

Si Clave = 3 entonces Importe = P3 * CantidadVendida Cantidad3 = Cantidad3 + CantidadVendida

Si Clave = 4 entonces Importe = P4 * CantidadVendida Cantidad4 = Cantidad4 + CantidadVendida

Si Clave = 5 entonces Importe = P5 * CantidadVendida Cantidad5 = Cantidad5 + CantidadVendida

4. Imprimir la Clave, la Cantidad Vendida y el Importe de la venta:

   Imprimir “Clave”: Clave

   Imprimir “Cantidad vendida”: CantidadVendida

   Imprimir "”Importe de la venta”: Importe

   Actualizar la recaudación total: RecaudacionTotal = RecaudacionTotal + Importe

   Preguntar si hay otra Venta: (Sí o No)

   Sí Venta=No

5. Imprimir la cantidad de boletos vendidos de cada tipo:

   Imprimir "Cantidad de boletos vendidos de la Clave 1:", Cantidad1

   Imprimir "Cantidad de boletos vendidos de la Clave 2:", Cantidad2

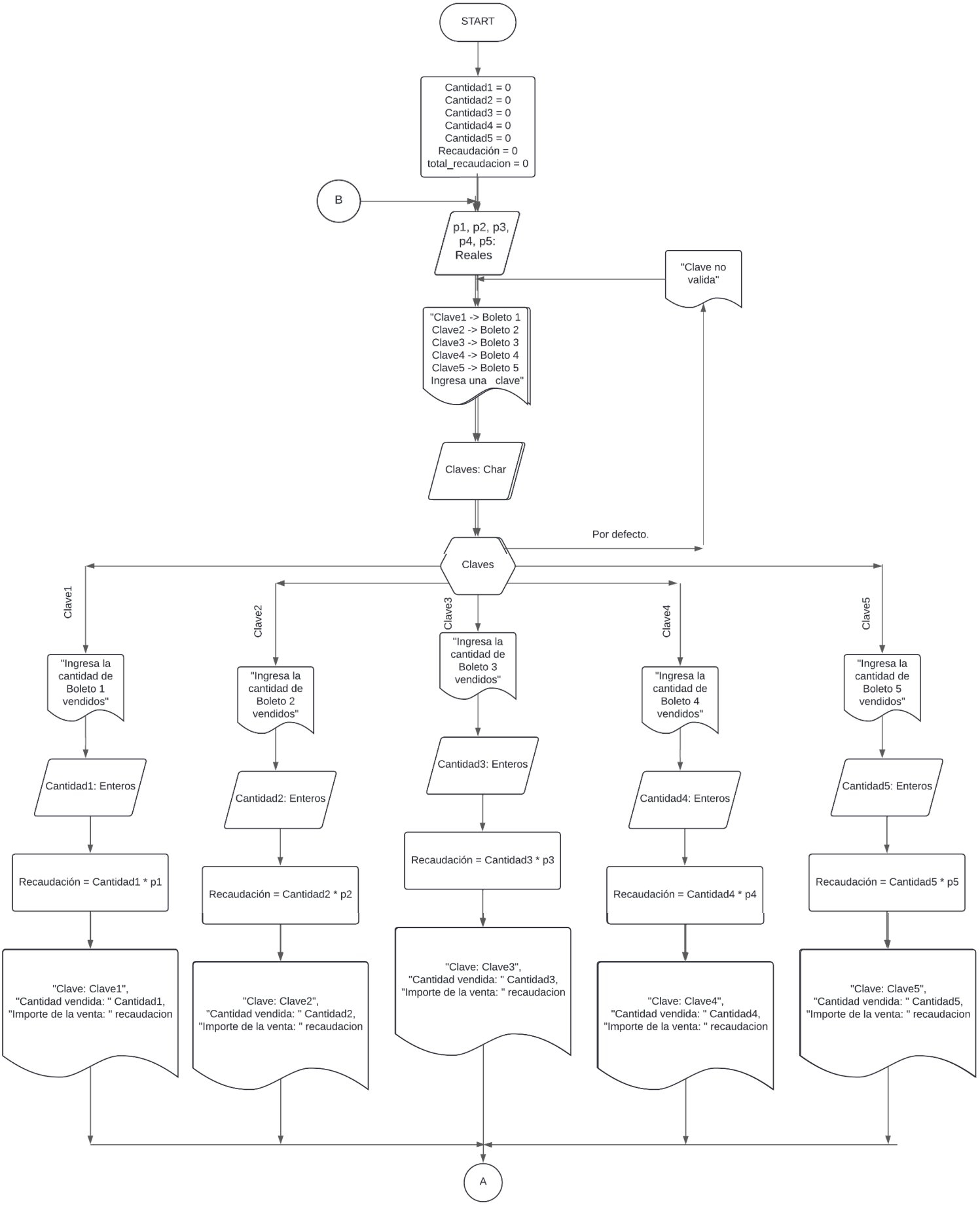
   Imprimir "Cantidad de boletos vendidos de la Clave 3:", Cantidad3

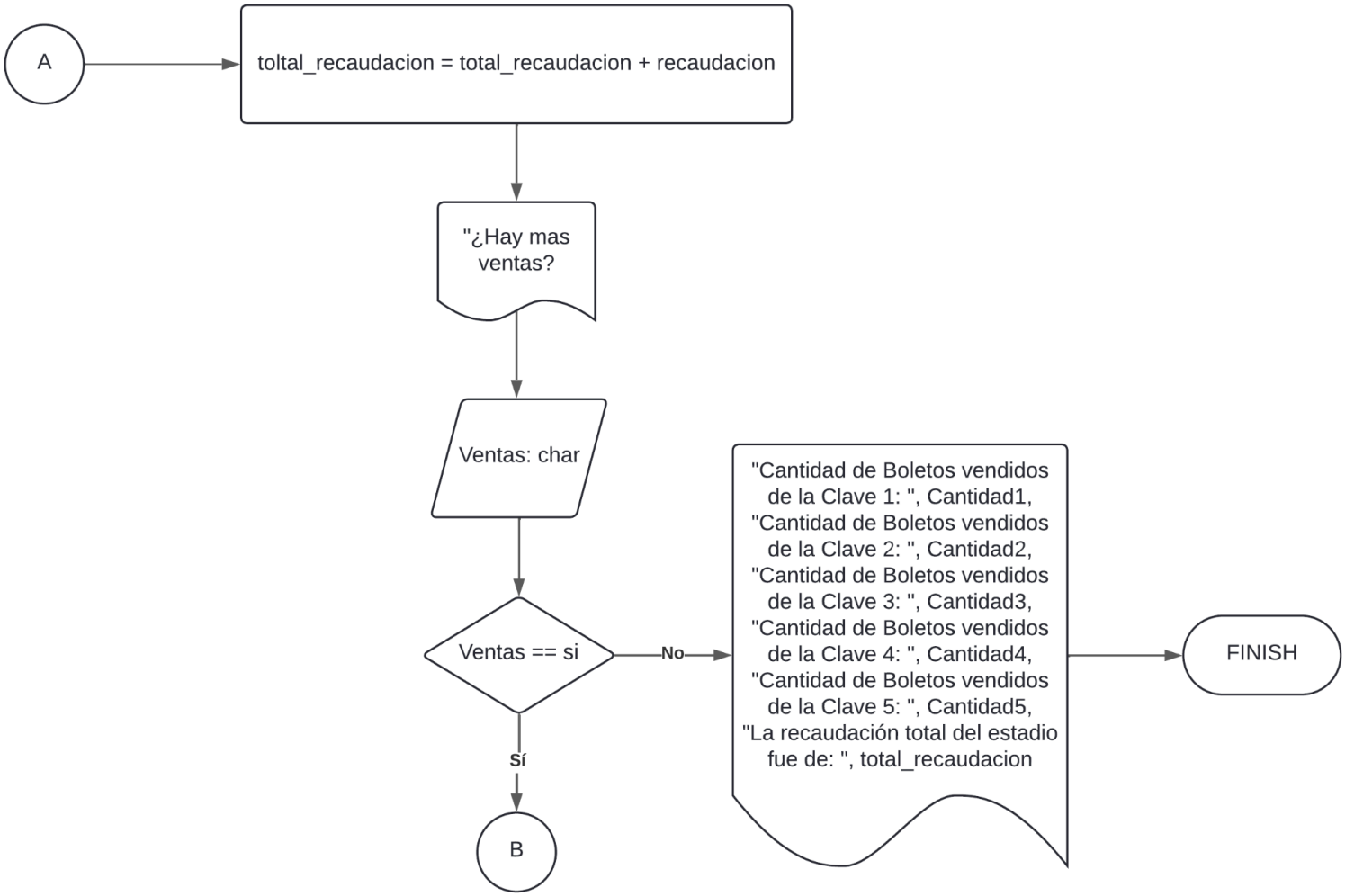
   Imprimir "Cantidad de boletos vendidos de la Clave 4:", Cantidad4

   Imprimir "Cantidad de boletos vendidos de la Clave 5:", Cantidad5

6. Imprimir la recaudación total del estadio: Imprimir “Recaudación total del estadio”: RecaudacionTotal
7. Fin del algoritmo

Diagrama de flujo:





Pseudocódigo:

INICIO Algoritmo_venta_boletos

// Inicialización de variables

int Cantidad1 = 0, Cantidad2 = 0, Cantidad3 = 0, Cantidad4 = 0, Cantidad5 = 0;

float p1, p2, p3, p4, p5; // Precios de los boletos

float Recaudacion = 0, total_recaudacion = 0;

char Clave, Ventas;

// Asignación de precios de boletos (p1, p2, p3, p4, p5)

p1 = ...; // Precio de boleto 1

p2 = ...; // Precio de boleto 2

```
p3 = ...; // Precio de boleto 3

p4 = ...; // Precio de boleto 4

p5 = ...; // Precio de boleto 5


do {

        // Entrada de la clave para seleccionar el tipo de boleto

        printf("Ingresa una clave (1 a 5 para los tipos de boletos): ");

        scanf(" %c", &Clave);


        // Selección del tipo de boleto basado en la clave

        switch (Clave) {

          case '1':

                printf("Ingresa la cantidad de boletos vendidos: ");

                scanf("%d", &Cantidad1);

                Recaudacion = Cantidad1 * p1;

                break;

          case '2':

                printf("Ingresa la cantidad de boletos vendidos: ");

                scanf("%d", &Cantidad2);

                Recaudacion = Cantidad2 * p2;

                break;

          case '3':

                printf("Ingresa la cantidad de boletos vendidos: ");

                scanf("%d", &Cantidad3);

                Recaudacion = Cantidad3 * p3;

                break;

          case '4':

                printf("Ingresa la cantidad de boletos vendidos: ");
```

```
                scanf("%d", &Cantidad4);
                Recaudacion = Cantidad4 * p4;
                break;
            case '5':
                printf("Ingresa la cantidad de boletos vendidos: ");
                scanf("%d", &Cantidad5);
                Recaudacion = Cantidad5 * p5;
                break;
            default:
                printf("Clave no válida.\n");
                break;
        }

        // Actualización de la recaudación total
        total_recaudacion += Recaudacion;

        // Preguntar si hay más ventas
        printf("¿Hay más ventas? (si/no): ");
        scanf(" %c", &Ventas);

    } while (Ventas == 's' || Ventas == 'S');

    // Imprimir el resumen de ventas
    printf("Cantidad de Boletos vendidos de la Clave 1: %d\n", Cantidad1);
    printf("Cantidad de Boletos vendidos de la Clave 2: %d\n", Cantidad2);
    printf("Cantidad de Boletos vendidos de la Clave 3: %d\n", Cantidad3);
    printf("Cantidad de Boletos vendidos de la Clave 4: %d\n", Cantidad4);
    printf("Cantidad de Boletos vendidos de la Clave 5: %d\n", Cantidad5);
```

```
// Imprimir la recaudación total
printf("La recaudación total del estadio fue de: %.2f\n", total_recaudacion);
```

Tabla 2: Prueba de escritorio de venta de boletos en un estadio.

Clave	Precio de Boleto	Cantidad Vendida	Importe de la venta
1	$50	100	$5000
2	$30	150	$4500
3	$20	50	$1000
4	$10	200	$2000
5	$5	300	$1500
		Recaudación total del estadio:	$14000

Ejercicio 3. Números Primos Menores a M.

Un número positivo N es un número primo si los únicos enteros positivos que lo dividen son exactamente 1 y N. Indique la forma en que dado un número M, se obtenga y cuente todo los números primos menores a M.

Algoritmo:

1. Se requiere encontrar todos los números primos menores que M y cuantos hay.
2. Definir una lista de números *primos* para almacenar los números primos encontrados .
3. Definir una variable *contador* e iniciarla en 0 para llevar el conteo de números primos.
4. Para cada número *n* desde 2 hasta M -1 (o ambos) verificar si *n* es primo.
5. Para cada n hacer lo siguiente:
    - Inicializar una variable en *esPrimo* en *Verdadero*.
    - Para cada número de *i* desde 2 hasta la raíz cuadrada de *n* (redondeada hacia abajo), hacer lo siguiente:
        - ❖ Si *n* es divisible por *i* (es decir, *n % i == 0*), entonces:
            - ➢ Establecer *esPrimo* en *Falso* (porque *n* tiene un divisor diferente de 1 y *n*).
            - ➢ Salir del bucle de verificación.

6. Si después de la verificación *esPrimo* sigue siendo *Verdadero*, agregar *n* a la lista primos y aumentar el *contador* en 1.
7. Al finalizar el bucle de iteración, la lista *primos* contendrá todos los números primos menores que M.
8. La variable *contador* indicará la cantidad total de números primos menores a M.
9. Mostrar o devolver a lista *primos*.
10. Mostrar o devolver el valor de *contador*.

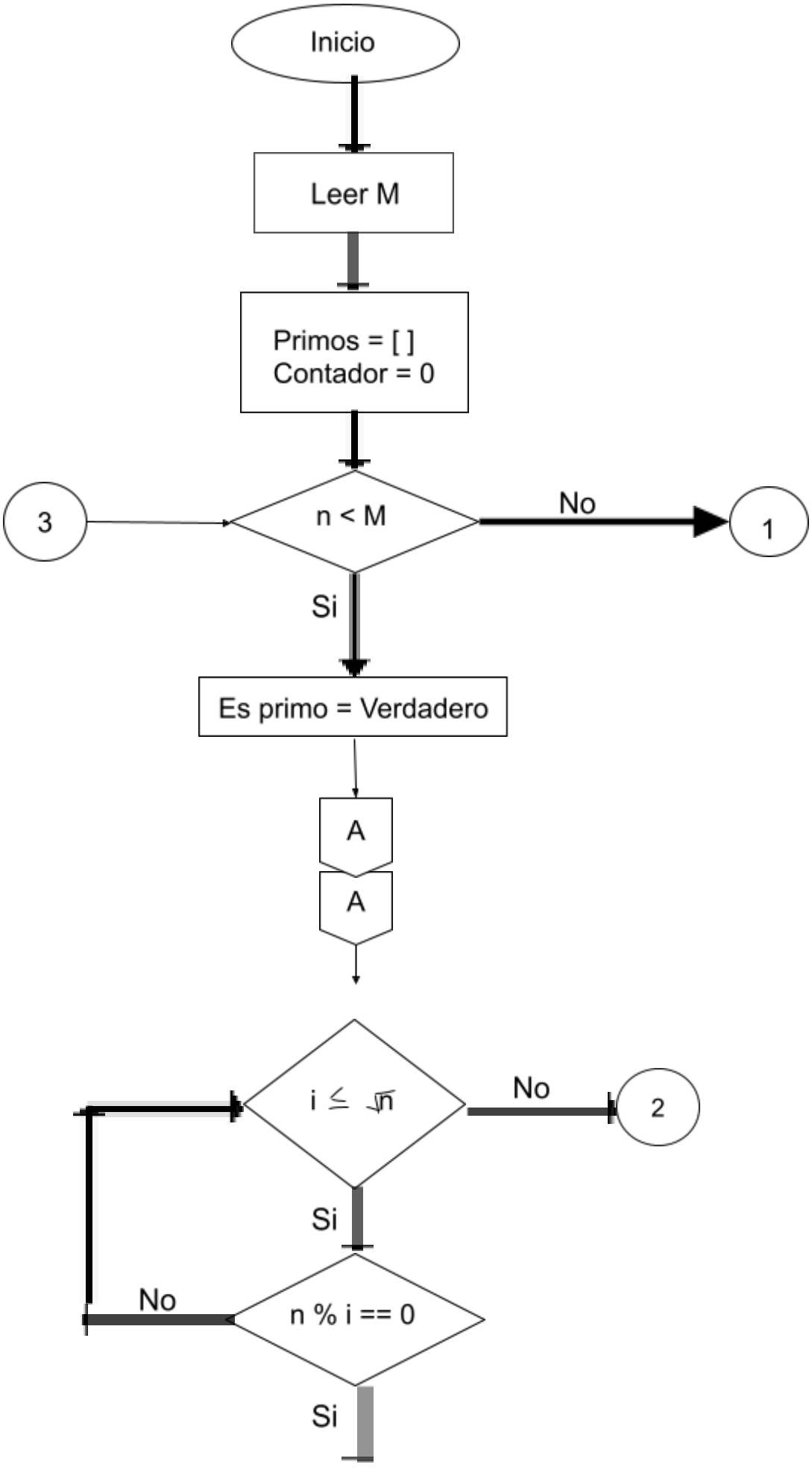
Ejemplo de Implementación de pseudocódigo.

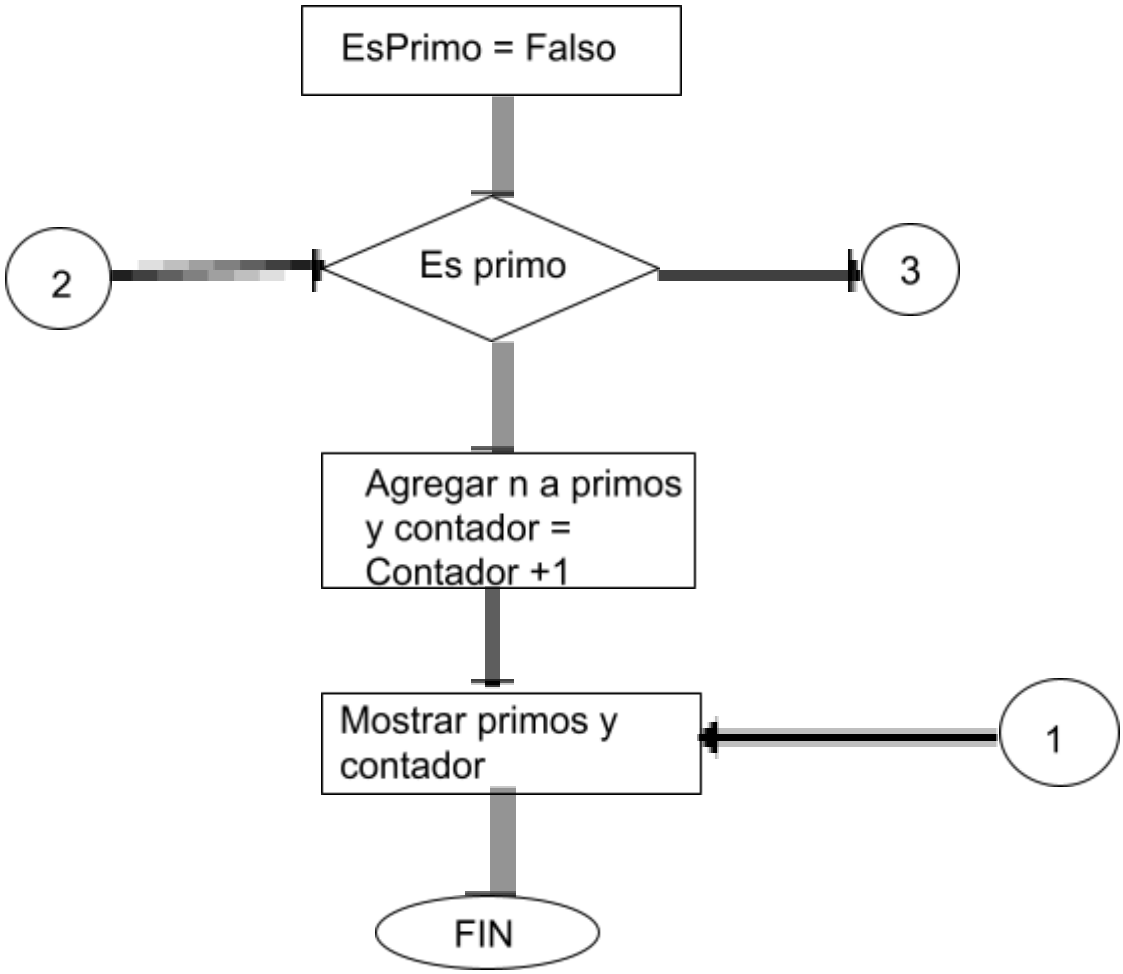
```
función contarPrimosMenoresQue(M):
    primos = lista vacía
    contador = 0
    para n desde 2 hasta M-1:
        esPrimo = Verdadero

        para i desde 2 hasta raíz cuadrada(n):
            si n % i == 0:
                esPrimo = Falso
                romper el bucle
        si esPrimo:
            agregar n a primos
    contador = contador + 1
    devolver (primos, contador)
```

Este algoritmo garantiza que todos los números primos menores a M sean encontrados y contados de manera eficiente

Diagrama de flujo:




EsPrimo = Falso
Es primo
2
3
Agregar n a primos
y contador =
Contador +1
Mostrar primos y
contador
1
FIN

Pseudocódigo:

```
INICIO
   LEER M
   contador <- 0

   PARA i DESDE 2 HASTA M-1 HACER
      primo <- VERDADERO
      PARA j DESDE 2 HASTA RAÍZ(i) HACER
         SI i MOD j = 0 ENTONCES
            primo <- FALSO
            SALIR DEL BUCLE
         FIN SI
      FIN PARA

      SI primo ENTONCES
         contador <- contador + 1
      FIN SI
   FIN PARA

   IMPRIMIR "Número de primos menores a M:", contador
FIN
```

Ejercicio 4: La sumatoria desde n=1 hasta N

Desarrollar el diagrama de flujo que permita realizar la sumatoria desde n=1 hasta N, donde N es el valor máximo que puede alcanzar la sumatoria. $\sum_{n=1}^{N} \left(\frac{-1^{n}}{n}\right)$

Algoritmo:

INICIO Algoritmo para la sumatoria desde n=1 hasta N

Declaración de variables:

N=REAL, i, suma

Escribir “Ingrese el valor de N”

Leer N

suma=0,

Para i=1 hasta N, hacer:

suma=suma+i

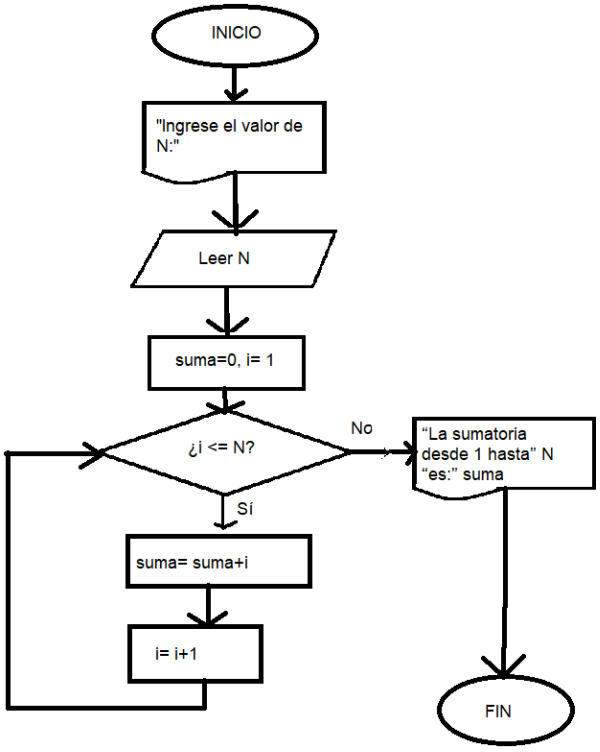
i= i+1

FIN Para

Escribir “La sumatoria desde 1 hasta” N “es:” suma

FIN

Diagrama de flujo:



Pseudocódigo:

Tabla 4. Prueba de escritorio de la sumatoria desde n=1 hasta N

N	suma=0 (suma=+i)	i=1 i=i+1
1	0+1= 1	1+1= 2
2	1+2= 3	2+1= 3
3	3+3= 6	3+1= 4
4	6+4= 10	4+1= 5
5	10+5 =15	5+1= 6

Conclusiones

1. ***Santiago Durán Rendón***: Se logró con éxito la implementación de pseudocódigos para resolver diferentes problemas algorítmicos. Cada uno de los ejercicios permitió reforzar habilidades clave en la lógica computacional, desde la manipulación de secuencias numéricas hasta la gestión de estructuras repetitivas y condicionales. El uso de pruebas de escritorio fue fundamental para validar la precisión de los algoritmos desarrollados.
2. ***Santiago Noriega Chiu***: Con los pseudocódigos se pudo reforzar el conocimiento de programación, así como reforzamos la manera de sintaxis correcta de este, mientras que la prueba de escritorio, ayuda a validar el uso de estos pseudocódigos, por lo que estos dos elementos son indispensables para el mundo de la programación, y así facilitarnos el desarrollo de un algoritmo.
3. ***José de Jesús Ramírez Reyes***: Crear pseudocódigos que representen soluciones algorítmicas con la sintaxis y semántica adecuadas es clave en el mundo de la programación. Estos pseudocódigos son como un plano que nos ayuda a visualizar cómo funcionará el algoritmo antes de escribir una sola línea de código. Cuando escribimos pseudocódigo, estamos desglosando el problema en pasos más simples y claros, para poder trabajar de forma efectiva.

Bibliografía

N/A