**Objetivo: Identificar elementos de entrada y resultados esperados de problemas planteados mediante diagramas de flujo**

**GA3-220501093-AA2-EV01**

****

Isidro J Gallardo Navarro

Ficha:3070299

 2025

Tecnología en Análisis y Desarrollo de Software.

ADSO

Lista de chequeo a cubrir:

●

1. Introducción

Este documento presenta el desarrollo de algoritmos fundamentales utilizando programación estructurada, con énfasis en la identificación precisa de datos de entrada, procesos de transformación y resultados esperados. La metodología se centra en la representación visual mediante diagramas de flujo y la implementación en pseudocódigo estructurado.

El algoritmo principal abordado corresponde al cálculo de edad actual de una persona a partir de su fecha de nacimiento y la fecha actual, demostrando el uso de estructuras de control y operaciones aritméticas con fechas.

2. Algoritmo Principal: Cálculo de Edad Actual

2.1 Análisis del Problema

Descripción: Desarrollar un algoritmo que determine la edad en años de una persona, dados su fecha de nacimiento y la fecha actual.

Datos de Entrada:

Fecha de nacimiento (día, mes, año)

Fecha actual (día, mes, año)

Datos de Salida:

Edad actual de la persona en años

Proceso: Calcular la diferencia entre las fechas considerando si ya cumplió años en el año actual.

2.2 Pseudocódigo Estructurado



2.3 Diagrama de Flujo

[INICIO]

|

[Leer dia\_nac, mes\_nac, año\_nac]

|

[Leer dia\_actual, mes\_actual, año\_actual]

|

[edad = año\_actual - año\_nac]

|

[mes\_actual < mes\_nac O (mes\_actual = mes\_nac Y dia\_actual < dia\_nac)?]

/ \

[Sí] [No]

| |

[edad = edad - 1] |

| |

\ /

|

[edad < 0?]

/ \

[Sí] [No]

| |

[Mostrar "Error"] [Mostrar edad]

| |

\ /

|

[FIN]

2.4 Prueba Funcional

Caso de Prueba 1:

Entrada: Nacimiento: 15/03/1995, Actual: 10/08/2024

Proceso: 2024 - 1995 = 29, ya cumplió años (marzo < agosto)

Salida: 29 años ✓

Caso de Prueba 2:

Entrada: Nacimiento: 20/12/1990, Actual: 10/08/2024Proceso: 2024 - 1990 = 34, no ha cumplido años (agosto < diciembre)

Salida: 33 años ✓

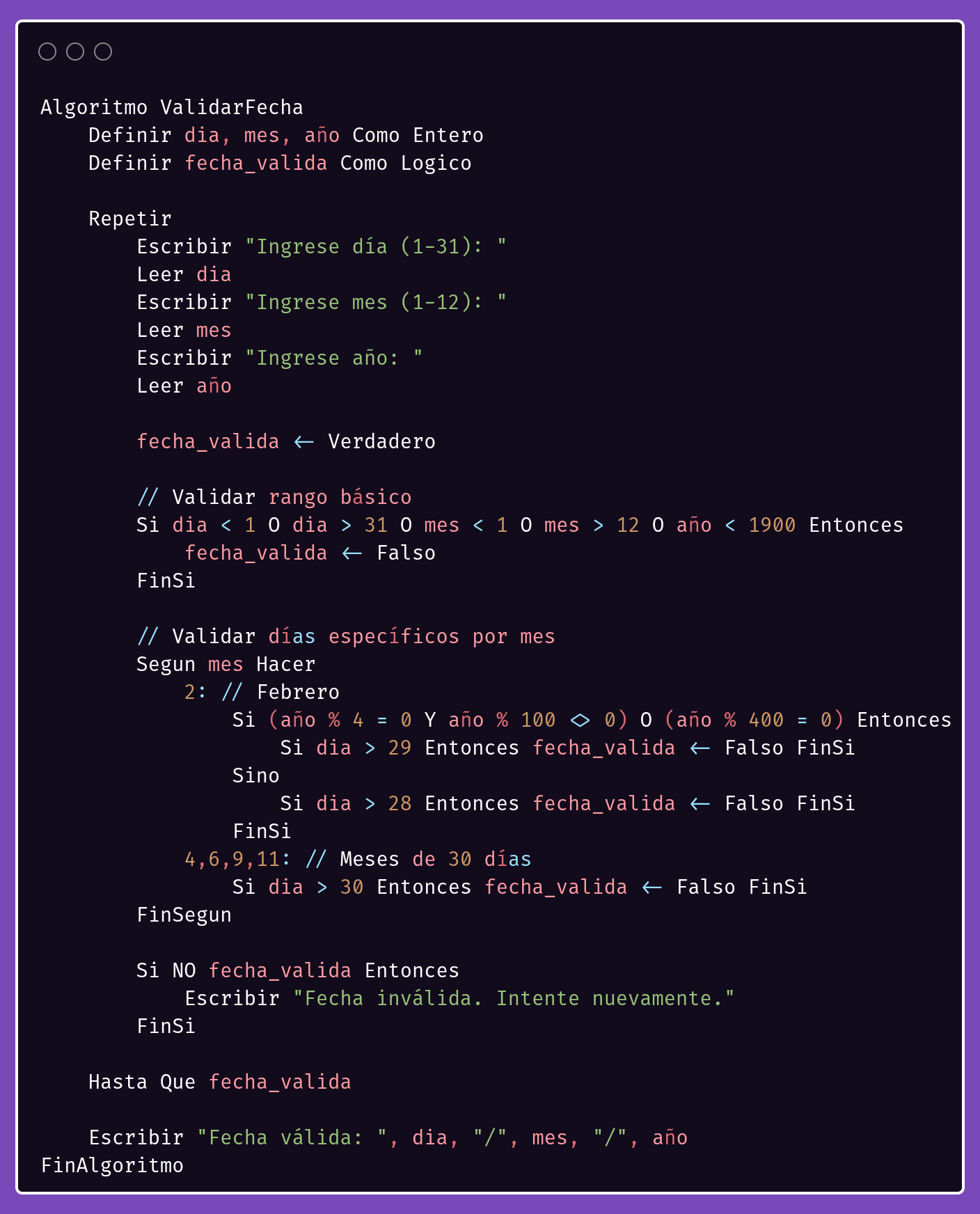
Caso de Prueba 3:

Entrada: Nacimiento: 10/08/2000, Actual: 10/08/2024

Proceso: 2024 - 2000 = 24, cumple años hoy

Salida: 24 años ✓

3. Algoritmos Complementarios con Estructuras Cíclicas

3.1 Algoritmo de Validación de Fechas

3.2 Diagrama de Flujo - Validación de Fechas

[INICIO]

|

[fecha\_valida = Falso]

|

[Leer dia, mes, año]

|

[fecha\_valida = Verdadero]

|

[dia<1 O dia>31 O mes<1 O mes>12 O año<1900?]

/ \

[Sí] [No]

| |

[fecha\_valida = Falso] |

| |

\ /

|

[mes = 2?]

/ \

[Sí] [No]

| |

[Año bisiesto?] |

/ \ |

[Sí] [No] |

| | |

[dia>29?] [dia>28?]|

| | |

\ / |

| |

[fecha\_valida = Falso si día inválido]

| |

\ /

|

[mes = 4,6,9,11?]

/ \

[Sí] [No]

| |

[dia > 30?] |

| |

[fecha\_valida = Falso] |

| |

\ /

|

[fecha\_valida?]

/ \

[No] [Sí]

| |

[Mostrar "Fecha [Mostrar fecha

inválida"] válida]

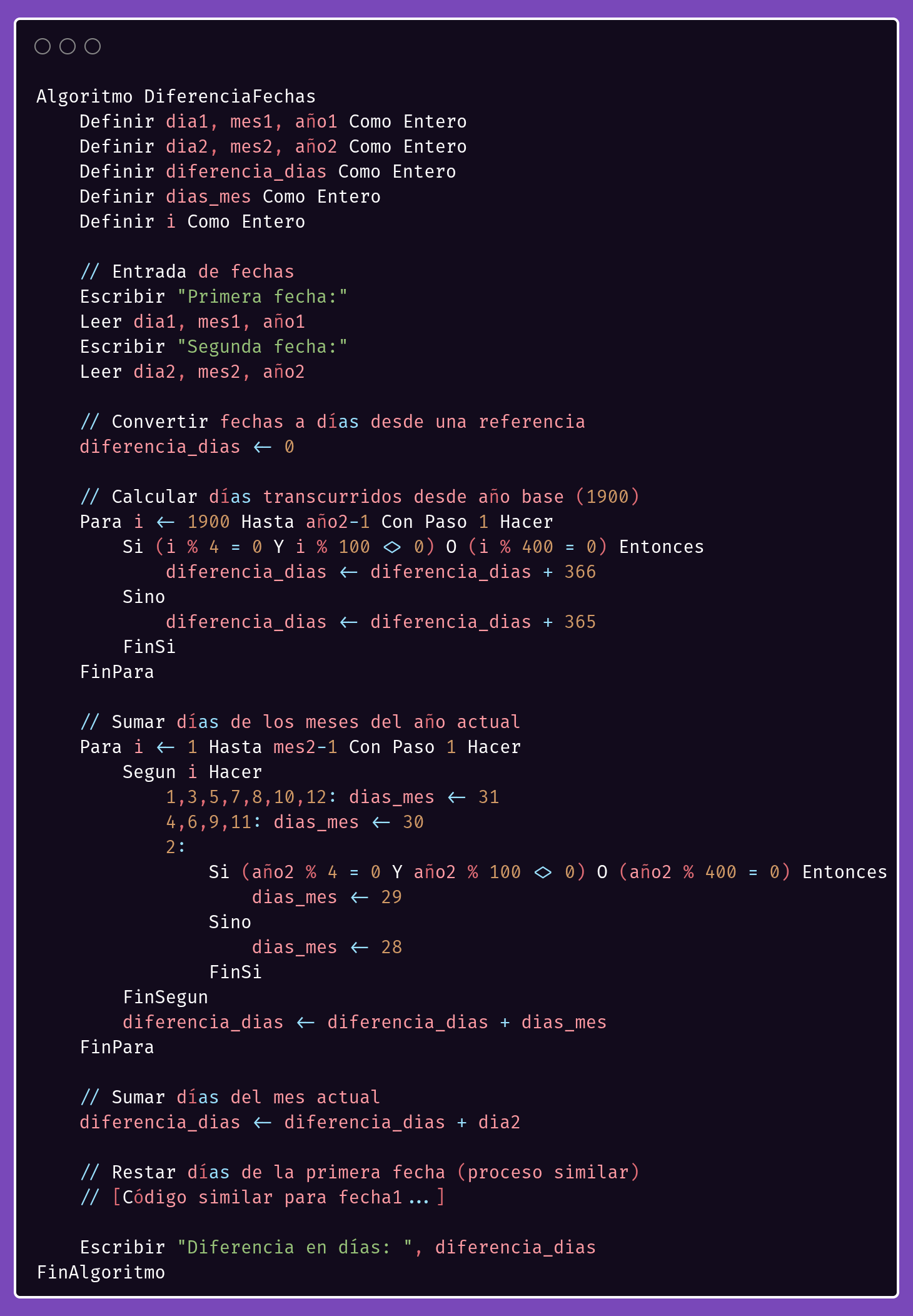
| |

\ /

|

[FIN]

### 3.3 Algoritmo de Cálculo de Diferencia de Fechas



4. Análisis de Estructuras de Control

4.1 Estructuras Secuenciales

Los algoritmos presentados utilizan estructuras secuenciales para:

Declaración de variables

Entrada de datos

Cálculos aritméticos básicos

Salida de resultados

4.2 Estructuras Condicionales

Estructura Si-Entonces-Sino:

Validación de fechas

Verificación de cumpleaños

Manejo de errores

Estructura Según-Hacer:

Validación de días por mes

Cálculo de días en meses específicos

4.3 Estructuras Cíclicas

Estructura Repetir-Hasta Que:

Validación de entrada de datos

Repetición hasta obtener datos válidos

Estructura Para:

Iteración sobre rangos de años

Cálculo acumulativo de días

4.4 Ventajas de la Programación Estructurada

Legibilidad: El código es fácil de leer y entender

Mantenibilidad: Facilita las modificaciones y correcciones

Reutilización: Los módulos pueden ser reutilizados

Depuración: Simplifica la localización de errores

Documentación: El flujo lógico es evidente

Conclusiones

La implementación de algoritmos para el cálculo de edad demuestra la aplicación práctica de los fundamentos de programación estructurada. El uso de diagramas de flujo facilita la comprensión del problema y la identificación de casos especiales.

Las estructuras cíclicas permiten implementar validaciones robustas, mientras que las estructuras condicionales manejan la lógica específica del dominio del problema. La metodología estructurada garantiza código mantenible, legible y verificable.

Herramientas recomendadas: PSeInt para desarrollo de pseudocódigo y generación automática de diagramas de flujo, facilitando la transición hacia lenguajes de programación específicos.