

Tema3. Diseño y realización de pruebas

IES PEDRO MERCEDES

CUENCA 2019/20

MARINA MORENO MARTÍNEZ

1. Introducción

- Las pruebas de software consisten en verificar y validar un producto software antes de su puesta en marcha.
- Constituyen una de las etapas del desarrollo de software, y básicamente consiste en probar la aplicación construida.
- Se integran dentro de las diferentes fases del ciclo de vida del software dentro de la ingeniería de software.

1. Introducción

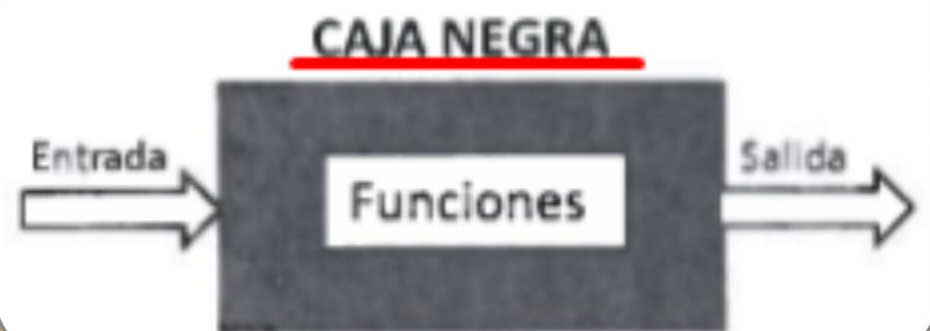
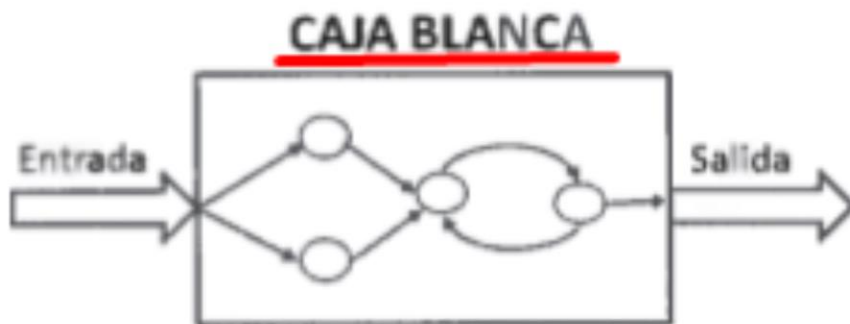
- Etapas de la ejecución de pruebas:
 - ✓ Planificación de pruebas
 - ✓ Diseño y construcción de los casos de prueba
 - ✓ Definición de los procedimientos de prueba
 - ✓ Ejecución de las pruebas
 - ✓ Registro de resultados obtenidos
 - ✓ Registro de errores encontrados
 - ✓ Depuración de los errores e informe de los resultados obtenidos.

2. Técnicas de diseño de pruebas

- Etapas de la ejecución de pruebas:
 - ✓ Planificación de pruebas
 - ✓ Diseño y construcción de los casos de prueba
 - ✓ Definición de los procedimientos de prueba
 - ✓ Ejecución de las pruebas
 - ✓ Registro de resultados obtenidos
 - ✓ Registro de errores encontrados
 - ✓ Depuración de los errores e informe de los resultados obtenidos.

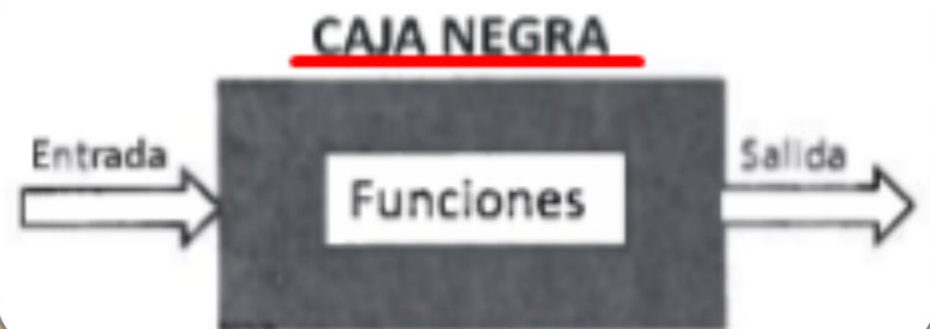
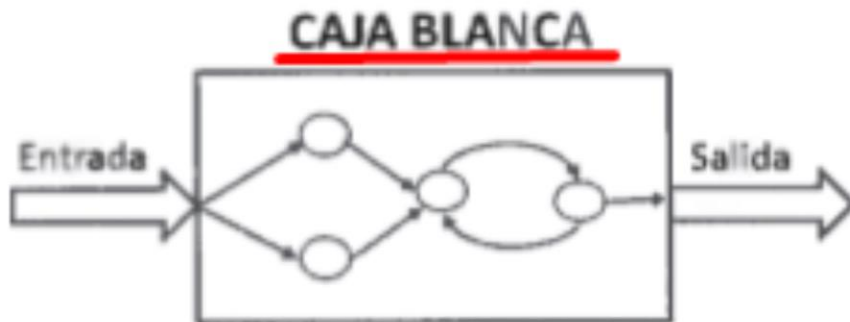
2. Técnicas de diseño de pruebas

- **Caso de prueba** conjunto de entradas, condiciones de ejecución y resultados esperados de salida para conseguir un objetivo particular o condición de prueba.
- Necesitamos **precondiciones** y **postcondiciones**.
- Técnicas o enfoques:



2. Técnicas de diseño de pruebas

- **Caso de prueba** conjunto de entradas, condiciones de ejecución y resultados esperados de salida para conseguir un objetivo particular o condición de prueba.
- Necesitamos **precondiciones** y **postcondiciones**.
- Técnicas o enfoques:



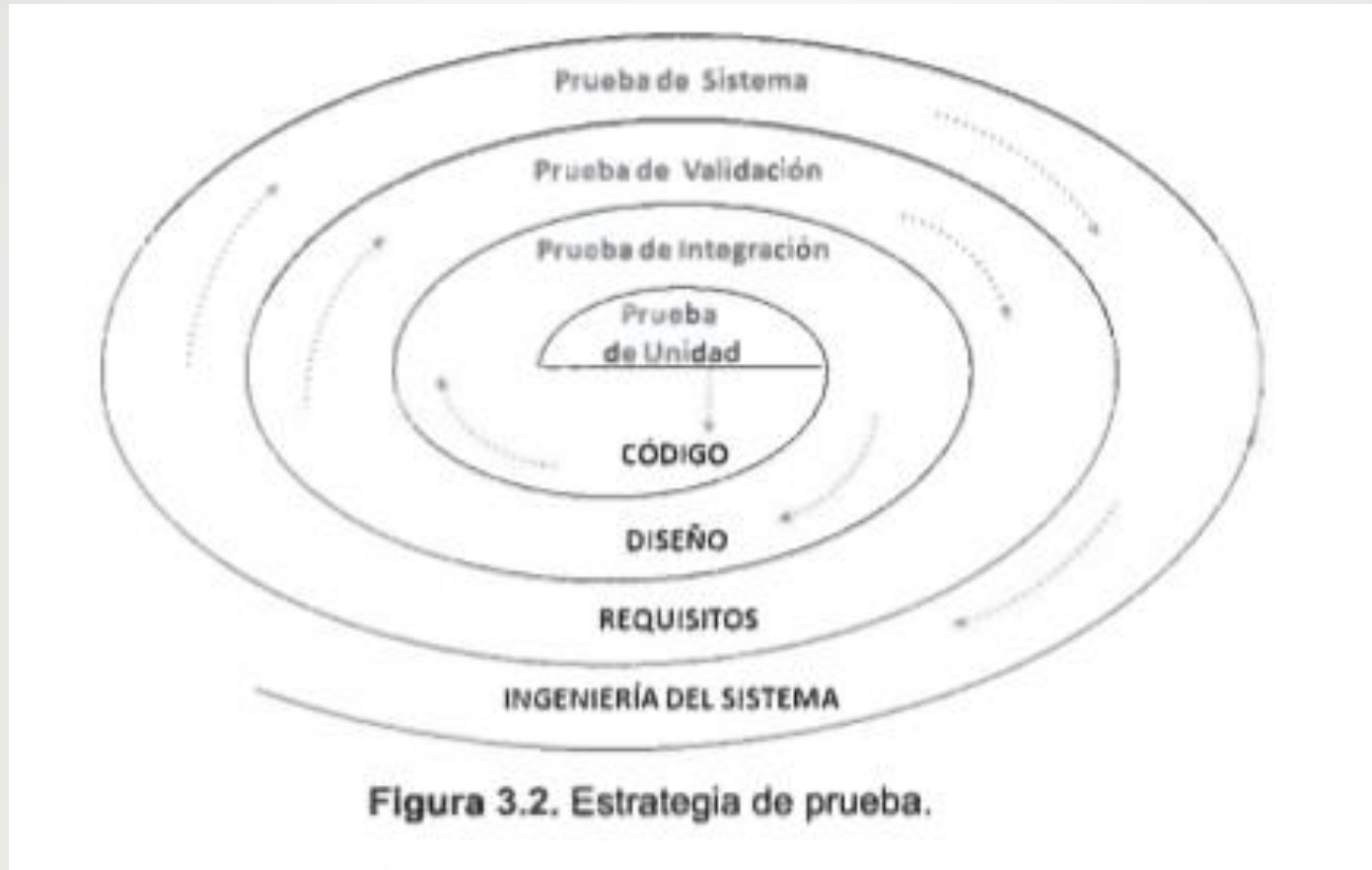
2.1 Pruebas de caja blanca

- **Pruebas estructurales o de caja de cristal**
- Se centran **en estudiar minuciosamente el código** de la aplicación.
- Casos de prueba que:
 - ✓ Garanticen que se ejecutan al menos una vez todos los caminos independientes de cada módulo.
 - ✓ • Ejecuten todas las sentencias al menos una vez.
 - ✓ • Ejecuten todas las decisiones lógicas en su parte verdadera y en su parte falsa.
 - ✓ • Ejecuten todos los bucles en SUS límites.
 - ✓ • Utilicen todas las estructuras de datos internas para asegurar su validez.
- Ejemplo: **Prueba del camino básico.**

2.1 Pruebas de caja negra

- También se les llama **prueba de comportamiento**.
- Se llevan a cabo sobre la interfaz del software.
- Comprueban que las salidas que devuelve la aplicación son las esperadas en función de las entradas que se proporcionen.
- Se intenta encontrar errores de las siguientes categorías:
 - ✓ Funcionalidades incorrectas o ausentes.
 - ✓ Errores de interfaz.
 - ✓ Errores en estructuras de datos en accesos a bases de datos externas.
 - ✓ Errores de rendimiento.
 - ✓ Errores de inicialización y finalización.
- Ejemplo: **Clases de equivalencia, análisis de valores límite.**

3. Estrategias de pruebas del Software



3. Estrategias de pruebas del Software

3.1 Prueba de unidad

- Se prueba cada unidad o módulo con el objetivo de eliminar errores en la interfaz y en la lógica interna.
- Se utilizan técnicas de caja negra y caja blanca.
- Se prueba:
 - ✓ La interfaz del módulo.
 - ✓ Las estructuras de datos locales.
 - ✓ Las condiciones límite.
 - ✓ Todos los caminos independientes de la estructura de control.
 - ✓ Todos los caminos de manejo de errores.

3. Estrategias de pruebas del Software

3.1 Prueba de unidad

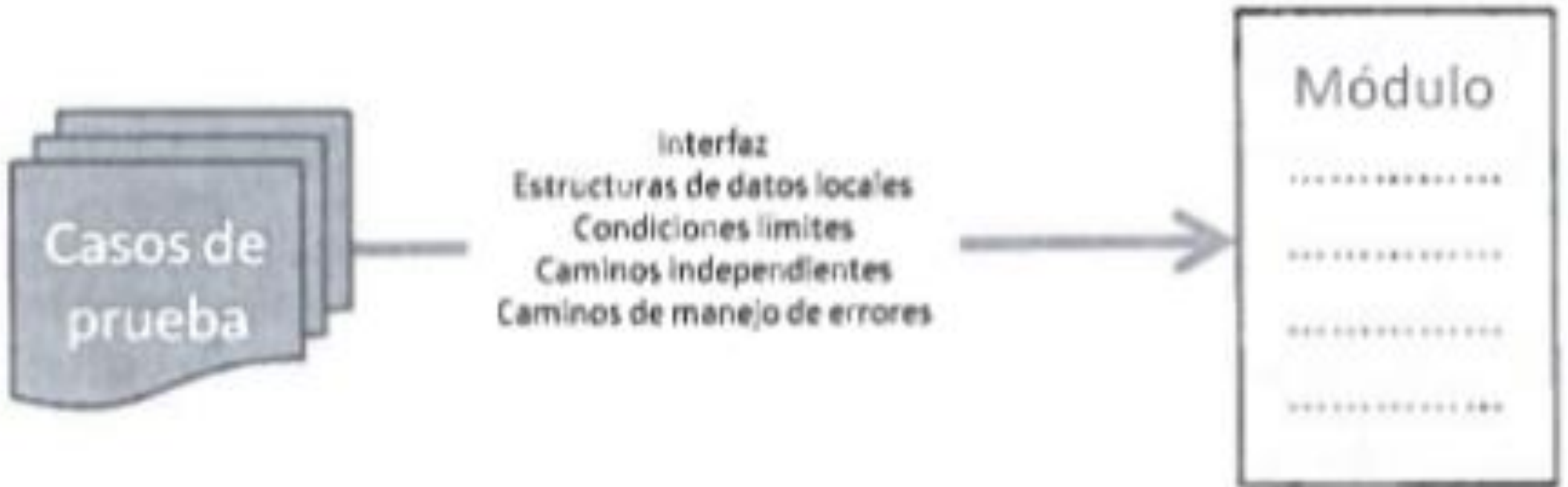


Figura 3.3. Prueba de unidad.

3. Estrategias de pruebas del Software

3.1 Prueba de integración

- Se prueba como interaccionan los distintos módulos.
- Hay dos enfoques fundamentales:
 - ✓ **Integración no incremental o big bang.**
 - ✓ **Integración incremental.** Estrategia ascendente y descendente.

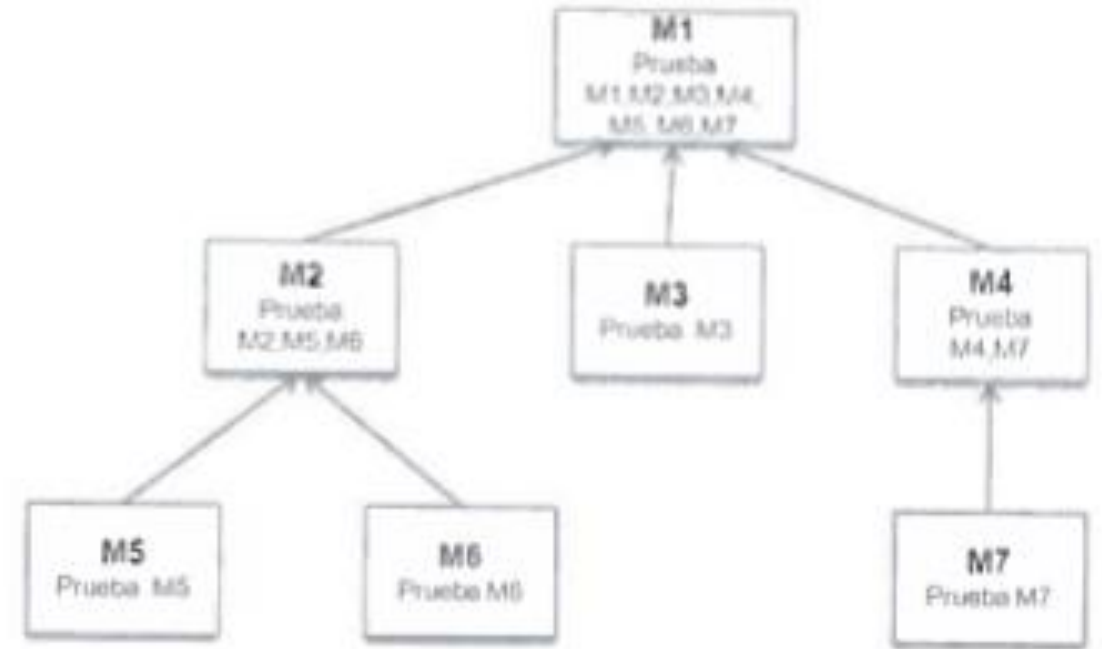


Figura 3.4. Prueba de integración Ascendente.

3. Estrategias de pruebas del Software

3.1 Prueba de validación

- La validación se consigue cuando el software funciona de acuerdo con las expectativas razonables del cliente definidas en el documento de especificación de requisitos del software o ERS.
- Se llevan a cabo una serie de pruebas de caja negra.
- Las técnicas utilizadas son:
 - ✓ **Prueba Alfa.** El cliente prueba el software en el lugar de desarrollo.
 - ✓ **Prueba Beta.** El cliente prueba el software en el lugar de trabajo.

3. Estrategias de pruebas del Software

3.1 Prueba de sistema

- La prueba del sistema está formada por un conjunto de pruebas cuya misión es ejercitar profundamente el software.
- Son las siguientes:
 - ✓ **Prueba de recuperación.**
 - ✓ **Prueba de seguridad.** Accesos ilegales.
 - ✓ **Prueba de resistencia (Stress).** Probar el programa bajo una gran demanda de recursos.

4. Documentación para las pruebas

- El estándar IEEE829-1998 describe el conjunto de documentos que pueden producirse durante el proceso de prueba.
- Son los siguientes:
 - ✓ **Plan de Pruebas.**
 - ✓ **Especificaciones de prueba** que contiene:
 - ✓ Diseño de las pruebas
 - ✓ Casos de prueba
 - ✓ Procedimientos de prueba
 - ✓ **Informes de Prueba.**

5. Pruebas de código

- La prueba del código consiste en la ejecución del programa(o parte de él)con el objetivo de encontrar errores.
- Se parte para su ejecución de un conjunto de entradas y una serie de condiciones de ejecución se observan y registran los resultados y se comparan con los resultados esperados.

5. Pruebas de código


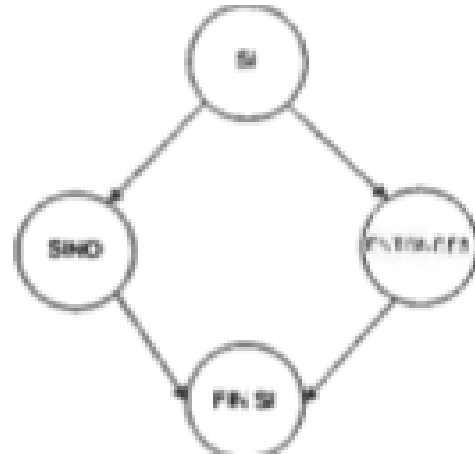
5.1 Prueba del camino básico

- Prueba de caja blanca.
- Obtienes la **complejidad ciclomática** de un proceso y a partir de ella se definen un conjunto básico de caminos de ejecución.
- Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se **ejecuta por lo menos una vez cada sentencia del programa.**
- Para la obtención de la medida de la complejidad lógica (o complejidad ciclomática) emplearemos una representación del flujo de control denominada **grafo de flujo o grafo del programa.**

5. Pruebas de código

5.1 Prueba del camino básico

- NOTACIÓN DE GRAFO DE FLUJO

ESTRUCTURA	GRAFO DE FLUJO
SECUENCIAL Instrucción 1 Instrucción 2 Instrucción n	
CONDICIONAL Si <condición> Entonces <Instrucciones> Si no <Instrucciones> Fin si	

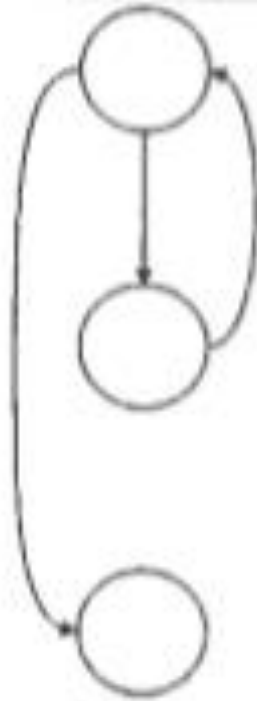
5. Pruebas de código

5.1 Prueba del camino básico

- NOTACIÓN DE GRAFO DE FLUJO

HACER MIENTRAS

Mientras <condición> Hacer
 <instrucciones>
Fin mientras



REPETIR HASTA

Repetir
 <instrucciones>
Hasta que <condición>



5. Pruebas de código

5.1 Prueba del camino básico

- NOTACIÓN DE GRAFO DE FLUJO

CONDICIONAL MÚLTIPLE

Según sea <variable> **Hacer**

Caso opción 1:

<Instrucciones>

Caso opción 2:

<Instrucciones>

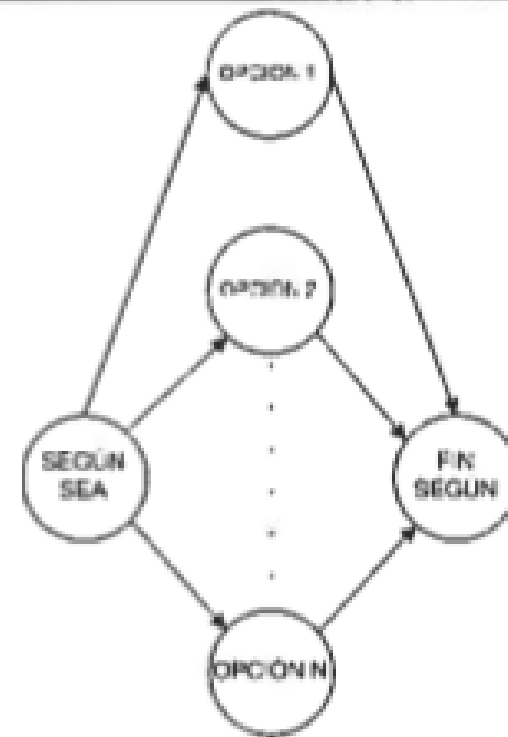
Caso opción 3:

<Instrucciones>

Otro caso:

<Instrucciones>

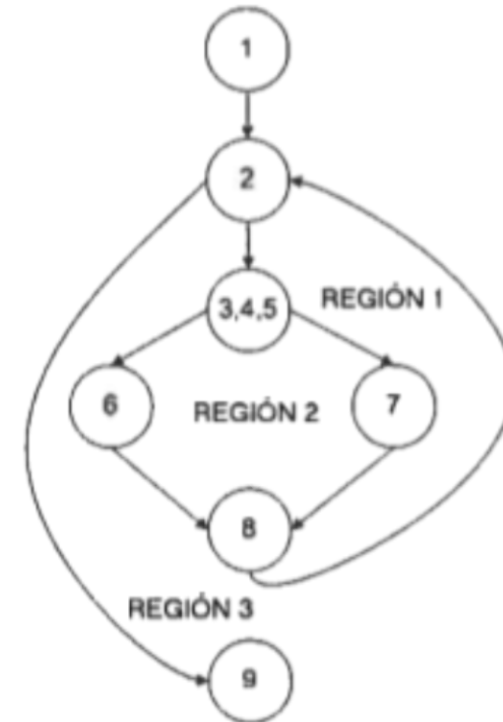
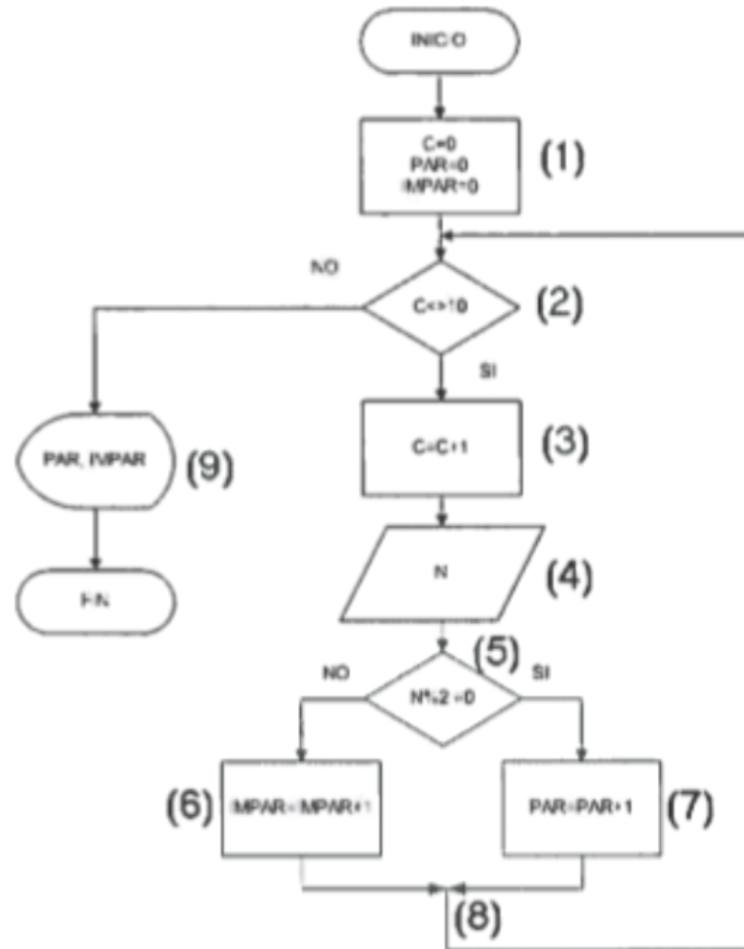
Fin según



5. Pruebas de código

5.1 Prueba del camino básico

- EJEMPLO:



5. Pruebas de código

5.1 Prueba del camino básico

- EJEMPLO 2:

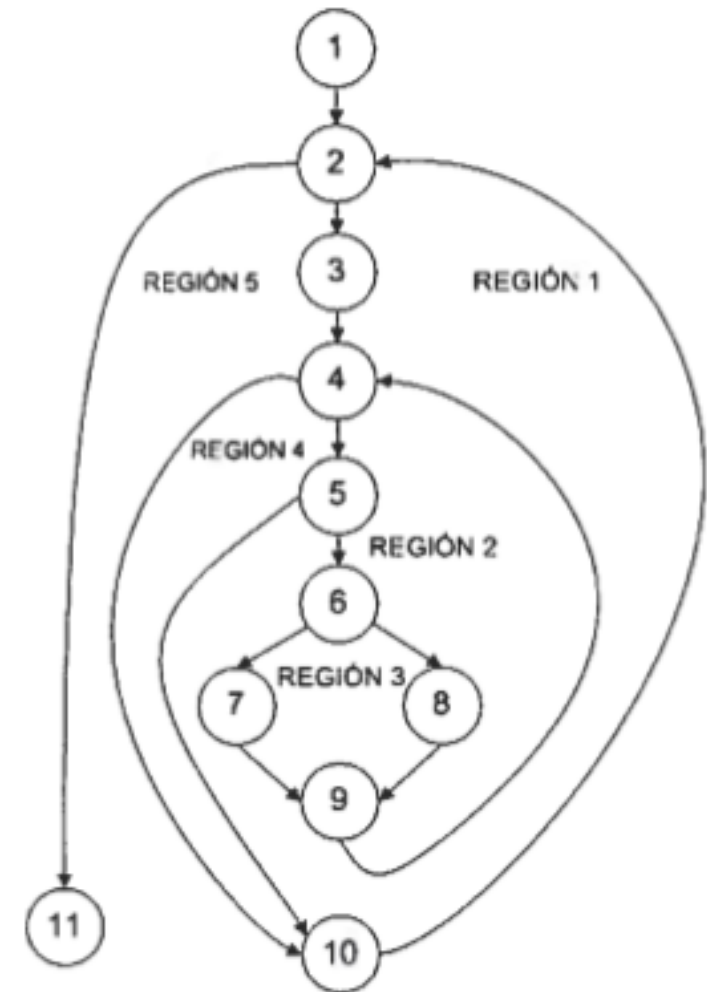
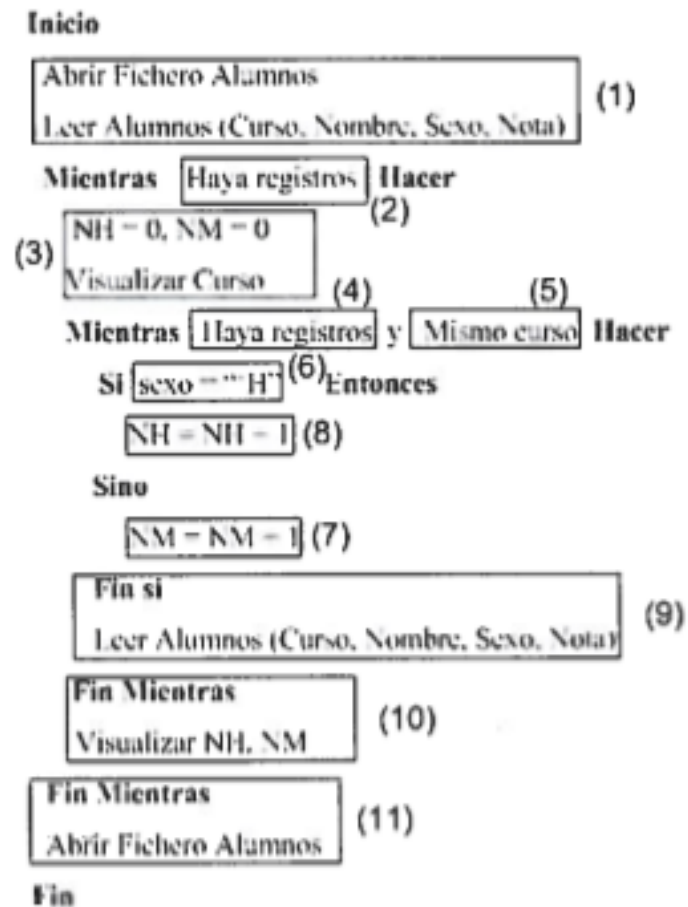


Figura 3.6. Pseudocódigo y grafo de flujo.

5. Pruebas de código

5.1 Prueba del camino básico

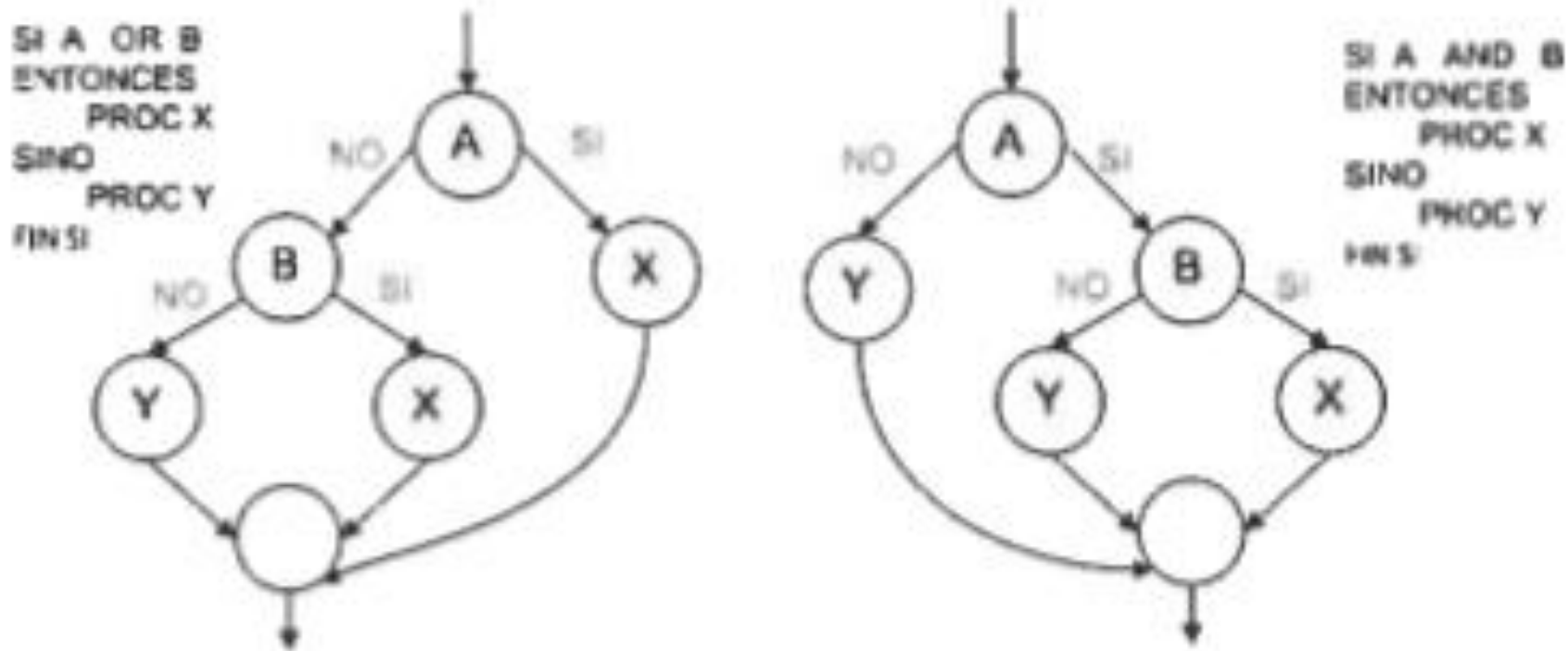


Figura 3.7. Lógica compuesta.

5. Pruebas de código

5.1 Prueba del camino básico

- **COMPLEJIDAD CILOMÁTICA**
- Es una métrica del software que proporciona una medida cuantitativa de la complejidad lógica de un programa.
- Establece **el número de caminos independientes** del conjunto básico de caminos de ejecución de un Programa, y por lo tanto, **el número de casos de prueba** que se deben ejecutar para asegurar que cada sentencia se ejecuta al menos una vez.
- Se representa por $V(G)$ y se puede calcular de tres formas:
 1. $V(G)$ = Número de regiones del grafo.
 2. $V(G)$ = Aristas — Nodos + 2.
 3. $V(G)$ = Nodos predicado + 1

5. Pruebas de código

5.1 Prueba del camino básico

Para el *Ejemplo 1*, la complejidad ciclomática es 3.

1. $V(G) = \text{Número de regiones del grafo} = 3.$
2. $V(G) = \text{Aristas} - \text{Nodos} + 2 = 8 - 7 + 2 = 3.$
3. $V(G) = \text{Nodos predicado} + 1 = 2 + 1 = 3$

Para el *Ejemplo 2*, la complejidad ciclomática es 5.

1. $V(G) = \text{Número de regiones del grafo} = 5.$
2. $V(G) = \text{Aristas} - \text{Nodos} + 2 = 14 - 11 + 2 = 5$
3. $V(G) = \text{Nodos predicado} + 1 = 4 + 1 = 5$

5. Pruebas de código

5.1 Prueba del camino básico

Se establecen los siguientes valores de referencia de la complejidad ciclomática:

Complejidad ciclomática	Evaluación de riesgo
Entre 1 y 10	Programas o métodos sencillos, sin mucho riesgo.
Entre 11 y 20	Programas o métodos más complejos, riesgo moderado.
Entre 21 y 50	Programas o métodos complejos, alto riesgo.
Mayor que 50	Programas o métodos no testeables, muy alto riesgo.

5. Pruebas de código

5.1 Prueba del camino básico

- **COMPLEJIDAD CILOMÁTICA**
- El valor de $V(G)$ nos da el número de caminos independientes del conjunto básico de un programa.
- Un **camino independiente** es cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una condición.
- En términos **del diagrama de flujo**, un camino independiente está constituido por lo menos por una arista que no haya sido recorrida anteriormente a la definición del camino.

5. Pruebas de código

5.1 Prueba del camino básico

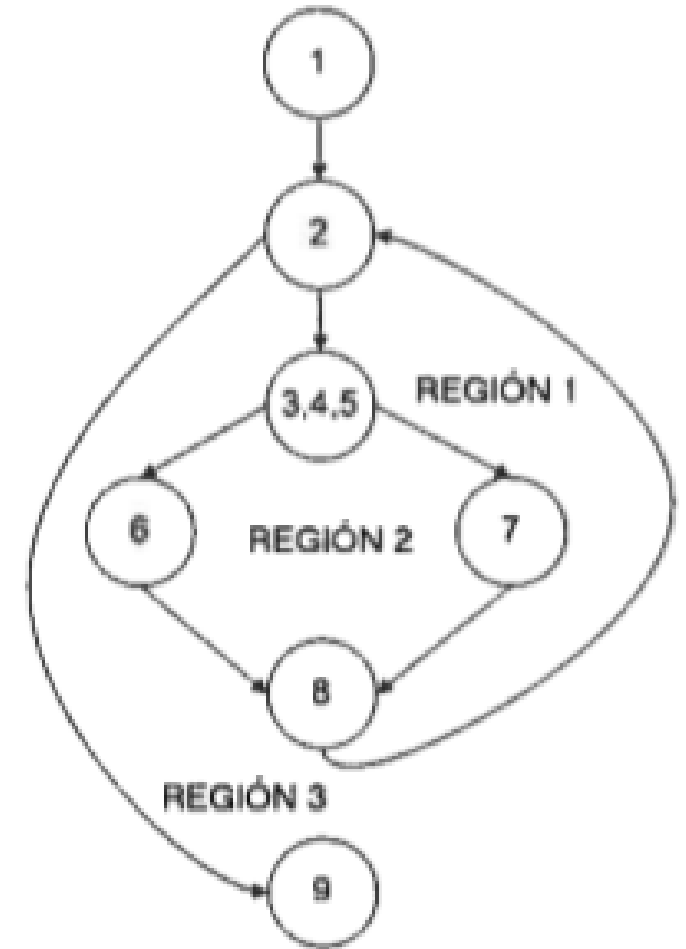
- COMPLEJIDAD CILOMÁTICA

Para el *Ejemplo 1*, la complejidad ciclomática es 3.

1. $V(G) = \text{Número de regiones del grafo} = 3.$
2. $V(G) = \text{Aristas} - \text{Nodos} + 2 = 8 - 7 + 2 = 3.$
3. $V(G) = \text{Nodos predicado} + 1 = 2 + 1 = 3$

Para el *Ejemplo 1*, un conjunto de caminos independientes será:

- Camino 1: 1 – 2 – 9
- Camino 2: 1 – 2 – 3, 4, 5 – 6 – 8 – 2 – 9
- Camino 3: 1 – 2 – 3, 4, 5 – 7 – 8 – 2 – 9



5. Pruebas de código

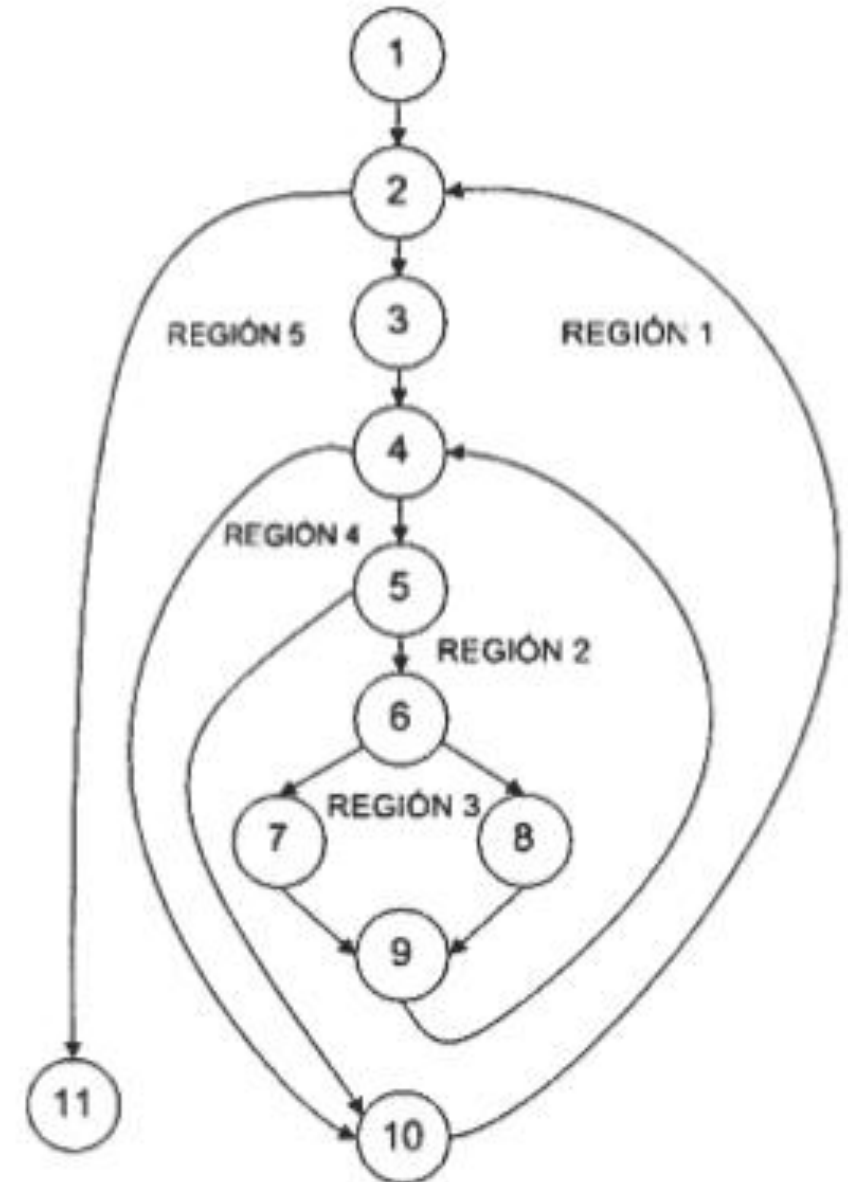
5.1 Prueba del camino básico

Para el *Ejemplo 2*, la complejidad ciclomática es 5.

1. $V(G) = \text{Número de regiones del grafo} = 5.$
2. $V(G) = \text{Aristas} - \text{Nodos} + 2 = 14 - 11 + 2 = 5$
3. $V(G) = \text{Nodos predicado} + 1 = 4 + 1 = 5$

Para el *Ejemplo 2*, un conjunto de caminos independientes será:

- Camino 1: 1 – 2 – 11
- Camino 2: 1 – 2 – 3 – 4 – 10 – 2 – 11
- Camino 3: 1 – 2 – 3 – 4 – 5 – 10 – 2 – 11
- Camino 4: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 4 – 10 – 2 – 11
- Camino 5: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 9 – 4 – 10 – 2 – 11



5. Pruebas de código

5.1 Prueba del camino básico

- **OBTENCIÓN DE CASOS DE PRUEBA**

Camino	Caso de prueba	Resultado esperado
1	Escoger algún valor de C tal que NO se cumpla la condición $C \neq 10$ $C = 10$	Visualizar el número de pares y el de impares
2	Escoger algún valor de C tal que SÍ se cumpla la condición $C \neq 10$. Escoger algún valor de N tal que NO se cumpla la condición $N \% 2 = 0$ $C = 1, N = 5$	Contar números impares
3	Escoger algún valor de C tal que SÍ se cumpla la condición $C \neq 10$. Escoger algún valor de N tal que SÍ se cumpla la condición $N \% 2 = 0$ $C = 2, N = 4$	Contar números pares

Camino 1: 1 - 2 - 9

Camino 2: 1 - 2 - 3, 4, 5 - 6 - 8 - 2 - 9

Camino 3: 1 - 2 - 3, 4, 5 - 7 - 8 - 2 - 9

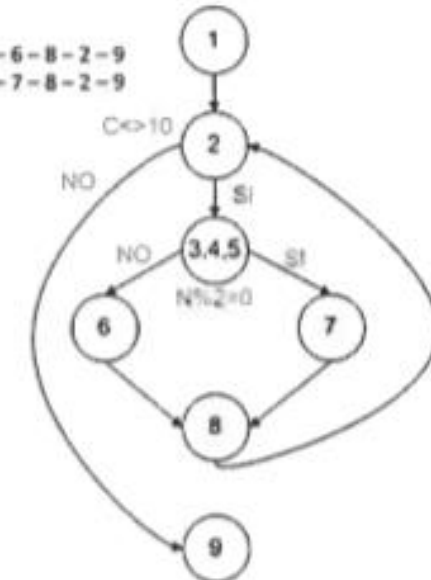


Figura 3.8. Nodos predichado del Ejemplo 1 con sus condiciones.

5. Pruebas de código

5.1 Prueba del camino básico

- **EJERCICIOS: 1, 4 Y 5**