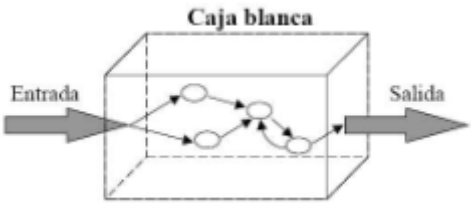
Apuntes de Contornos

Notas Básicas *Iago González Borines 1º DAW*

Pruebas de caja Blanca

Se centran en validar la estructura interna del programa y las
segundas se centran en validar los requisitos funcionales sin fijarse en el funcionamiento interno del programa (comprueban que el código produce el resultado esperado).



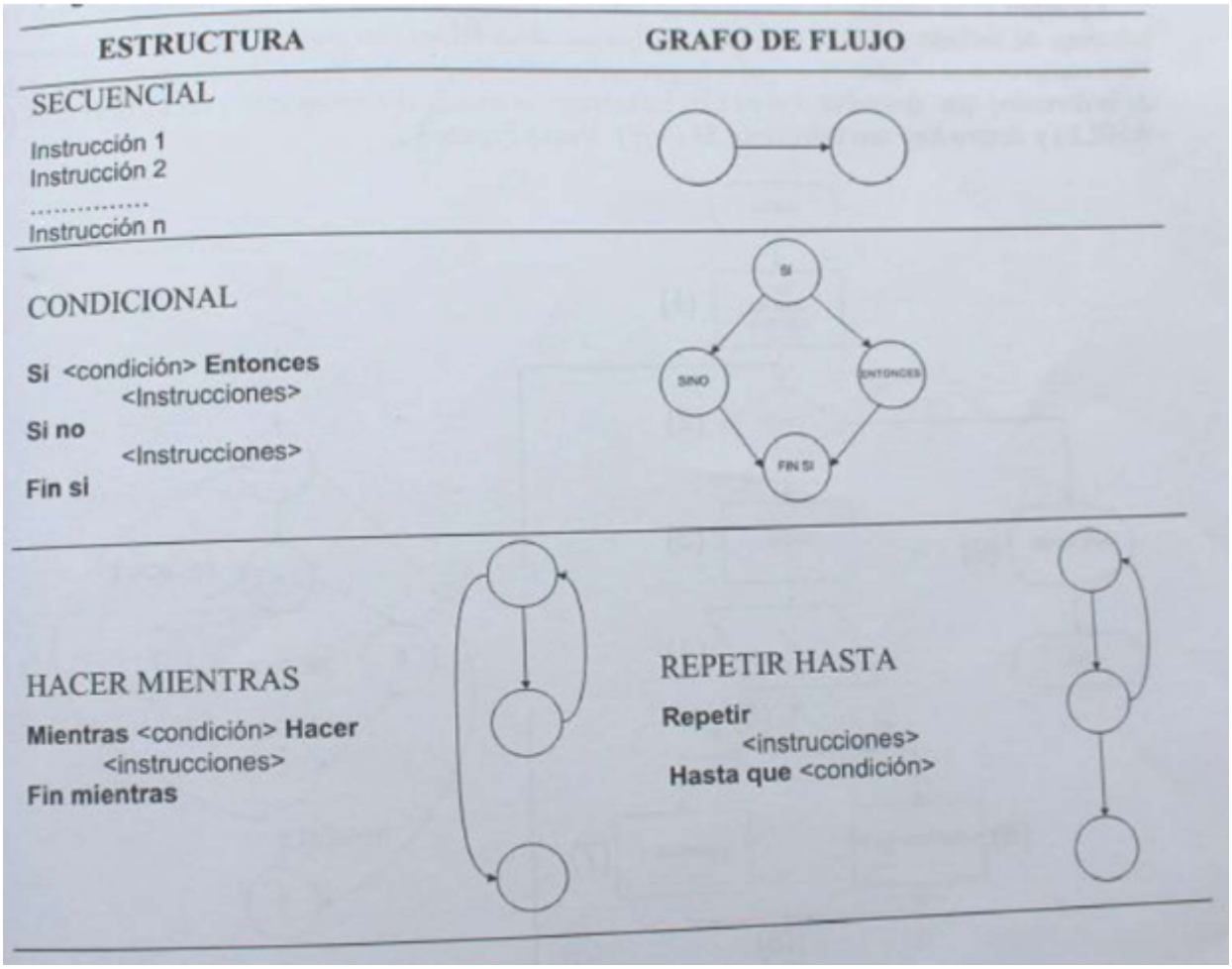
Pruebas de Cámino Básico

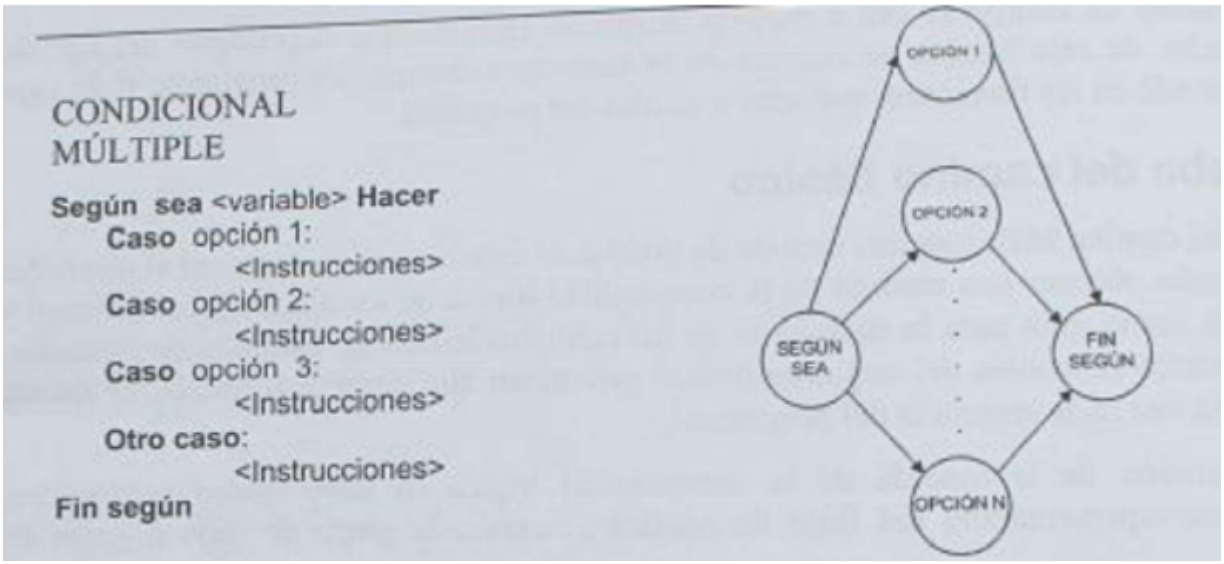
Estas valen principalmente para analizar todas las rutas que utiliza el programa durante su ejecución en distintas condiciones

Los pasos:

1. Dibujar el grafo.

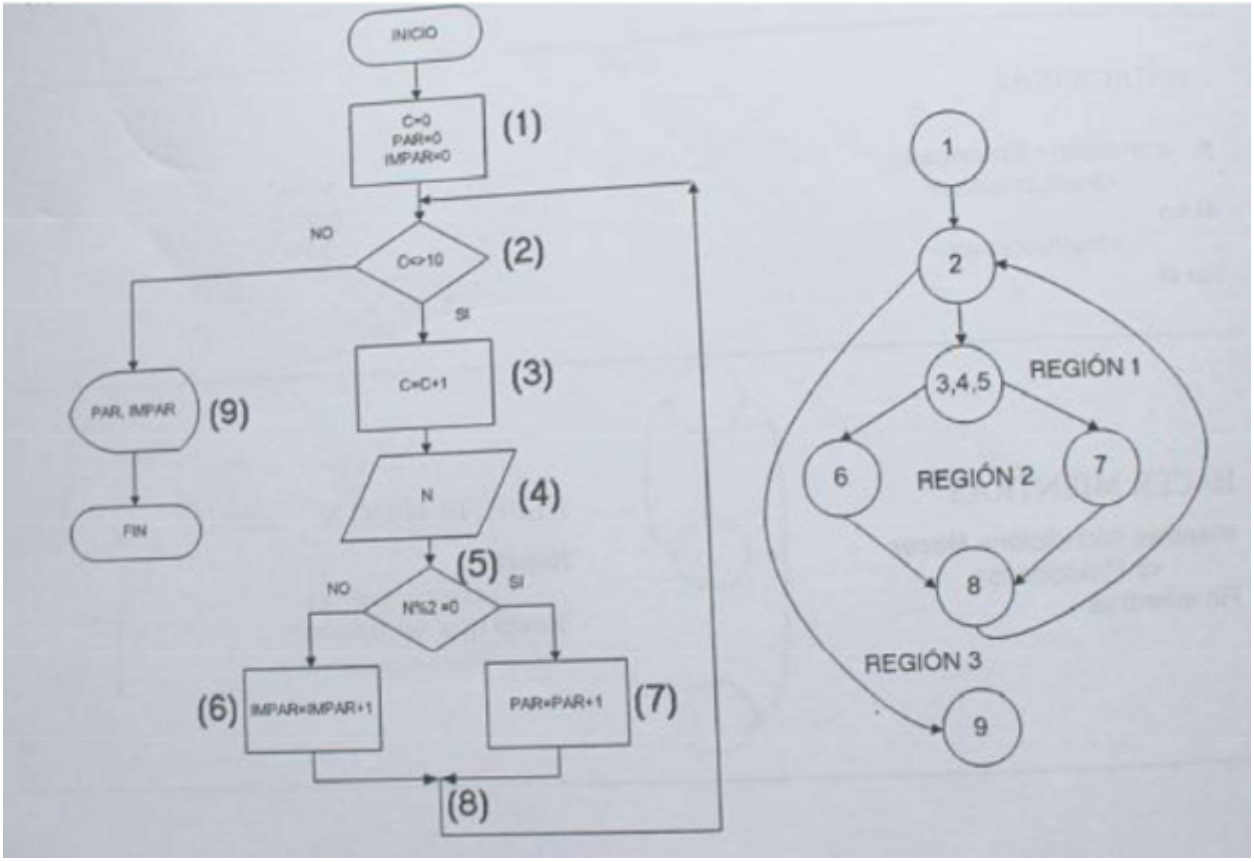
Para dibujar el grafo es necesario identificar las secuencias, las estructuras condicionales y las estructuras de repetición de código.





Ejemplo:

Se muestra el diagrama de flujo y el grafo de flujo para un programa que lee 10 números desde el teclado y muestra cuántos de los números leídos son pares y cuántos son impares. Para comprobar si el número es par o impar utilizamos el operador % de Java (devuelve el resto de la división) que devuelve 0 si es par. La estructura principal corresponde a un MIENTRAS (o WHILE) y dentro hay una estructura SI (o IF).



Términos

- Nodo: Es como se llema a los círculos que representan un paso.
- Aristas o enlaces: Básicamente lo que une los círculos.
- Régiones: Son áreas delimitadas por el propio grafo.
- Nodod predicado: Básicamente un nodo de condición del cual salen dos aristas.

2. Calcular la complejidad_ciclomática (número de caminos básicos existente).

Se trata de una medida para determinar la complejidad de un software y se calcula dfe la siguiente forma

$$M = E - N + 2$$

Siendo:

- M: La propia complejidad ciclomática
- E: es el número de aristas (conexiones entre nodos) en el diagrama de flujo.
- N: es el número de nodos (bloques de acción) en el diagrama de flujo.

Se puede calcular de otras formas como...

- M = Número de regiones del grafo = 3.
- M = Nodos predicado + 1

3. Encontrar los caminos_básicos.

Los caminos básicos son una técnica de prueba que se utiliza para asegurarse de que un programa de software se ha probado adecuadamente. La técnica de caminos básicos implica la identificación de todos los caminos posibles a través de un programa y la creación de pruebas para cada uno de estos caminos.

Para hacer estas pruebas tenemos que identificar que rutas sigue el programa en el grafo

Ejemplo:

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

4. Diseñar un caso_de_prueba para cada uno de los caminos.

Los casos de prueba son un conjunto de datos de entrada y salida que se utilizan para probar un programa de software. Los casos de prueba se crean para asegurarse de que el programa funcione correctamente en diferentes situaciones.

Para plasmar estos datos lo mejor es haciendo una tabla poniendo las entradas y salidas de cada uno de los caminos

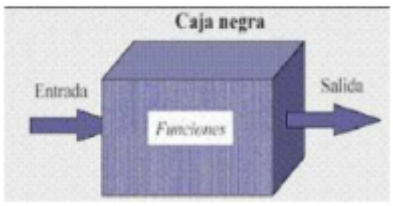
Ejemplo:

Camino	Caso de prueba	Resultado esperado
1	Escoger algún valor de C tal que NO se cumpla la condición C<>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<>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<>10. Escoger algún valor de N tal que SÍ se cumpla la condición N % 2 = 0 C=2, N = 4	Contar números pares

---

Pruebas de caja Negra

Estas se centran principalmente en el resultado del programa sin ver el proceso. Se trata de probar, si las salidas que devuelve la aplicación, o parte de ella, son las esperadas, en función de los parámetros de entrada que le pasemos. No nos interesa la implementación del software, solo si realiza las funciones que se esperan de él.



Partitiones equivalentes:

Esta técnica se utiliza para dividir los datos de entrada en grupos que se comportan de manera similar. Por ejemplo, si un programa de software toma un número como entrada, podemos dividir los números en grupos como números positivos, números negativos y el número cero. De esta forma, podemos crear casos de prueba para cada grupo y asegurarnos de que el programa funcione correctamente para cada uno de ellos.

Las pruebas de entrada que sabemos que nos van a dar error en el programa las llamamos **Clases no válidas**, y las que sabemos que deberían darnos un resultado las llamamos **Clases Válidas**.

Hay distintos tipos de clases de equivalencia y debemos usarlas par comprobar un rango alto de opciones:

1. Si el campo debe de **encontrarse en un rango**, se define una clase de equivalencia válida(dentro del rango) y dos no válidas(los límites inferiores y superiores).

2. Si el campo requiere de una **entrada específica**, se define una clase de equivalencia válida y dos no válidas.

3. Si el campo específica a **un elemento de un conjunto**, se define una clase de equivalencia válida por cada uno de los miembros del conjunto y una no válida.

4. Si el campo especifica **una condición de entrada lógica**, se define una clase de equivalencia válida y una no válida.

Condiciones de entrada	Nº de Clases de equivalencia válidas	Nº de Clases de equivalencia no válidas
1. Rango	1 CLASE VÁLIDA Contempla los valores del rango	2 CLASES NO VÁLIDAS Un valor por encima del rango Un valor por debajo del rango
2. Valor específico	1 CLASE VÁLIDA Contempla dicho valor	2 CLASES NO VÁLIDAS Un valor por encima Un valor por debajo
3. Miembro de un conjunto	1 CLASE VÁLIDA Una clase por cada uno de los miembros del conjunto	1 CLASE NO VÁLIDA Un valor que no pertenece al conjunto
4. Lógica	1 CLASE VÁLIDA Una clase que cumpla la condición	1 CLASE NO VÁLIDA Una clase que no cumpla la condición

Pasos:

1. Identificar las clases de equvalencia aplicando las opciones anteriores

Ejemplo:

Condición de entrada	Clase de equivalencia	Clases válidas	ID	Clases no válidas	ID
Código postal	Lógica (puede o no rellenarse)	En blanco	V1	No son dígitos	NV1
	Valor	Sólo números de 5 dígitos	V2	Números de menos de 5 dígitos	NV2
				Números de más de 5 dígitos	NV3

2. Definimos los casos de prueba, para ello se le debe haber asignado un ID a cada clase de quivalnecia para asi poder relacionar de una manera más organizada:

Ejemplo:

CASO DE PRUEBA	Clase de equivalencia	Entrada	Resultado esperado
C1	V1	En blanco	valor correcto
C2	V2	12345	valor correcto
C3	NV1	"largo"	Error, no son dígitos
C4	NV2	1234	Error, longitud errónea
C5	NV3	123456	Error, longitud errónea

Y ala ya estaría.

Análisis de valores límite

esta técnica se utiliza para probar los límites del programa, es decir, los valores mínimos y máximos que puede manejar.

Por ejemplo, si un programa de software toma un número entre 1 y 100 como entrada, podemos probar los valores límite de 1 y 100 para asegurarnos de que el programa funciona correctamente en estos casos extremos.

Pasos:

1. Identifica las variables de entrada: determina qué variables de entrada están involucradas en la función o programa que deseas probar.
2. Identifica los valores límite: identifica los valores máximos y mínimos que puede tomar cada variable de entrada.
3. Determina los valores justo dentro de los límites: determina los valores justo dentro de los límites para cada variable de entrada. Por ejemplo, si el valor mínimo es 1 y el valor máximo es 10, los valores justo dentro de los límites serían 2 y 9.
4. Determina los valores justo fuera de los límites: determina los valores justo fuera de los límites para cada variable de entrada. Por ejemplo, si el valor mínimo es 1 y el valor máximo es 10, los valores justo fuera de los límites serían 0 y 11.
5. Crea los casos de prueba: crea casos de prueba utilizando los valores límite y los valores justo dentro y justo fuera de los límites para cada variable de entrada. Asegúrate de probar todas las combinaciones posibles de valores límite y justo dentro y justo fuera de los límites.(ósea mándale una tablita ahí del chill u know).