

5 PRUEBAS DE CÓDIGO

La prueba de código consiste en la ejecución del programa 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. Se observará si el comportamiento del programa es el previsto o no y por qué.

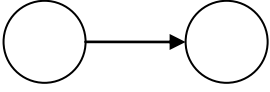
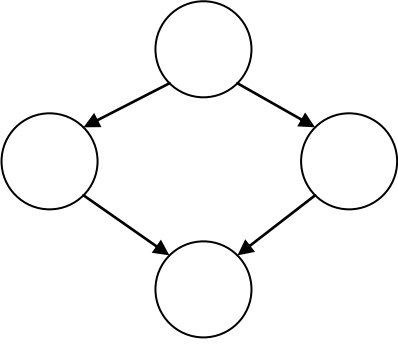
Para las pruebas de código se van a mostrar diferentes técnicas que dependerán del tipo de enfoque utilizado: de caja blanca, se centran en la estructura interna del programa; o de caja negra, más centrado en las funciones, entradas y salidas del programa.

5.1 Prueba del camino básico

La prueba del camino básico es una técnica de prueba de caja blanca que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de 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.

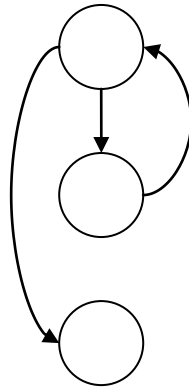
Para la obtención de la medida de la complejidad o ciclomática, emplearemos una representación del flujo de control denominada grafo de flujo o grafo de programa y realizaremos los siguientes pasos:

NOTACIÓN DE GRAFO DE FLUJO

ESTRUCTURA	GRAFO DE FLUJO
SECUENCIAL instrucción 1 instrucción 2 instrucción n	
CONDICIONAL SI <condición> ENTONCES <instrucciones> SINO <instrucciones> FINSI	

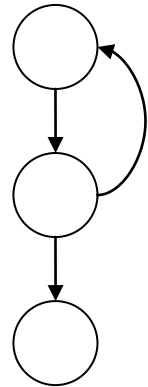
HACER MIENTRAS

Mientras <condición>
 hacer
 <instrucciones>
Fin_mientras



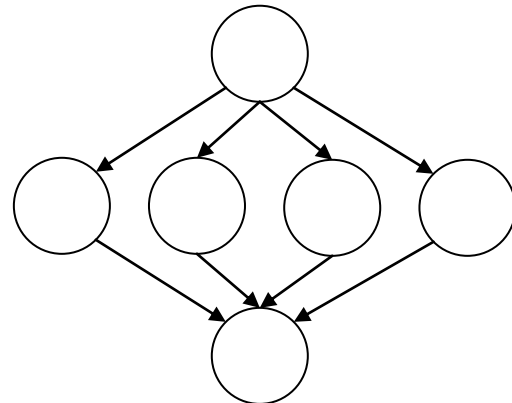
REPETIR HASTA

Repetir
 <instrucciones>
Hasta que <condición>



CONDICION MULTIPLE

Según sea <variable> **hacer**
 Caso opción 1:
 <instrucciones>
 Caso opción 2:
 <instrucciones>
 Caso opción 3:
 <instrucciones>
 Otro Caso:
 <instrucciones>
FINSegún



- Cada círculo del grafo se llama nodo y representa una o más sentencias, sin bifurcaciones, en pseudocódigo o código fuente.
- Las flechas del grafo de flujo se denominan aristas o enlaces y representan el flujo de control. Una arista termina en un nodo, aunque el nodo no tenga ninguna sentencia procedimental.
- Al nodo que contiene una condición se le llama nodo predicado y se caracteriza porque de él salen dos o más aristas.

COMPLEJIDAD CICLOMÁTICA

La complejidad ciclomática es una métrica del software que proporciona una medida cuantitativa de la complejidad lógica de un programa. En el contexto del método de prueba del camino básico, la complejidad ciclomática establece el número de caminos independientes del conjunto básico de caminos de ejecución de un programa, y por tanto, el número de casos de prueba que se deben ejecutar para asegurar que cada sentencia se ejecuta al menos una vez.

La complejidad ciclomática $V(G)$ puede calcularse de tres formas:

$$V(G) = \text{Número de regiones del grafo}$$

$$V(G) = \text{Aristas} - \text{Nodos} + 2$$

$$V(G) = \text{Nodos predicado} + 1$$

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.

OBTENCIÓN DE LOS CASOS DE PRUEBA

EL último paso de la prueba del camino básico es construir los casos de prueba que fuerzan la ejecución de un camino. Con el fin de comprobar cada camino, debemos escoger los casos de prueba de forma que las condiciones de los nodos predicado están adecuadamente establecidas.

5.2 Partición o clases de equivalencia

La partición equivalente es un método de prueba de caja negra que divide los valores de los campos de entrada de un programa en clases de equivalencia. Para identificar las clases de equivalencia se examina cada condición de entrada (son parte del dominio de valores de la entrada y normalmente son una frase en la especificación) y se divide en dos o más grupos. Se definen dos tipos de clases de equivalencia:

Clases válidas: son los valores válidos.

Clases no válidas: son los valores no válidos.

Las clases de equivalencia se definen según una serie de directrices:

1. Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos no válidas.
2. Si una condición de entrada especifica un valor específico, se define una clase de equivalencia válida y dos no válidas.
3. Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una no válida.
4. Si una condición de entrada es lógica, se define una clase de equivalencia válida y una no válida.

Para representar las clases de equivalencia para cada condición de entrada se puede usar una tabla. En cada fila se definen las clases de equivalencia para la condición de entrada, se añade un código a cada clase definida (válida y no válida) para usarlo en la definición de los casos de prueba.

A partir de esta tabla se generan los casos de prueba. Utilizamos las condiciones de entrada y las clases de equivalencia (a las que se asignó un código en la columna COD, también se podría haber asignado un número a cada clase). Los representamos en otra tabla donde cada fila representa un caso de prueba con los códigos de las clases de equivalencia que se aplican, los valores asignados a las condiciones de entrada y el resultado esperado según el enunciado del problema.

Al rellenar la tabla de casos de prueba se han tenido en cuenta estas dos reglas: los casos de prueba válidos cubren tantas clases de equivalencia válidas como sea posible y los casos de prueba no válidos cubren una sola clase no válida.

Los casos de prueba se van añadiendo a la tabla hasta que todas las clases de equivalencia válidas y no válidas hayan sido cubiertas.

5.3 Análisis de valores límite

El análisis de valores límite se basa en que los errores tienden a producirse con más probabilidad en los límites o extremos de los campos de entrada.

Esta técnica complementa a la anterior y los casos de prueba elegidos ejercitan los valores justo por encima y por debajo de los márgenes de la clase de equivalencia. Además, no solo se centra en las condiciones de entrada, sino que también se exploran las condiciones de salida definiendo las clases de equivalencia de salida.

Las reglas son las siguientes:

1. Si una condición de entrada especifica un rango de valores, se deben diseñar casos de prueba para los límites del rango y para los valores justo por encima y por debajo del rango.
2. Si una condición de entrada especifica un número de valores, se deben diseñar casos de prueba que ejerciten los valores máximo, mínimo, un valor justo por encima del máximo y un valor justo por debajo del mínimo.
3. Aplicar la regla 1 para la condición de salida.
4. Usar la regla 2 para la condición de salida.
5. Si las estructuras de datos internas tienen límites preestablecidos, hay que asegurarse de diseñar casos de prueba que ejercite la estructura de datos en sus límites, primer y último elemento.