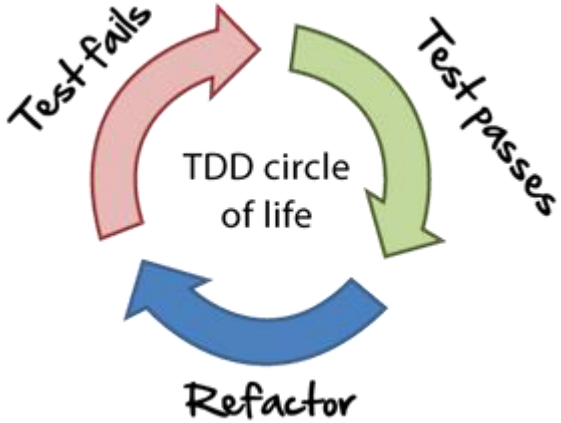
Florida

1º DAM/DAW EDE

U6. Diseño de pruebas de software

4 - Clasificación según el enfoque

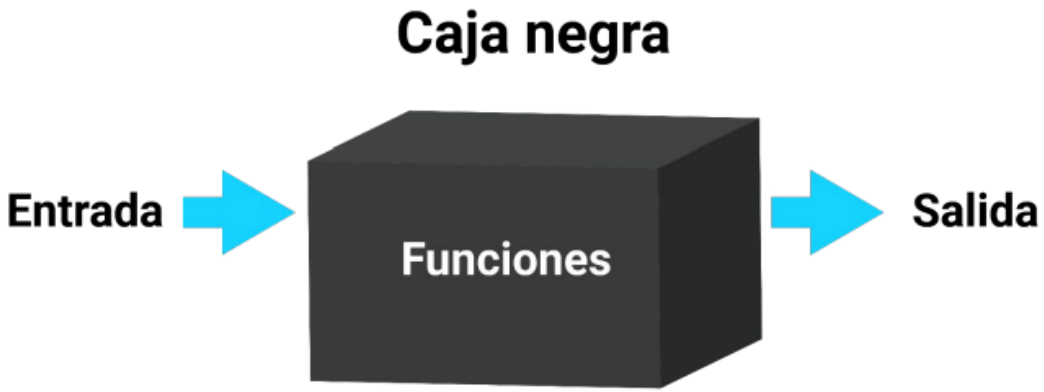


Clasificación general. Criterios

- **Según el objetivo:**
  - Pruebas funcionales.
  - Pruebas no funcionales.
- **Según el enfoque:**
  - Pruebas de caja negra.
  - Pruebas de caja blanca.
- **Según el método:**
  - Pruebas manuales.
  - Pruebas automáticas.

Pruebas de caja negra

- Se entiende el **código como una caja opaca**, inaccesible.
- Se realizan sin conocer el detalle del código a evaluar.
- **Únicamente** se tienen en cuenta **entradas y salidas**.

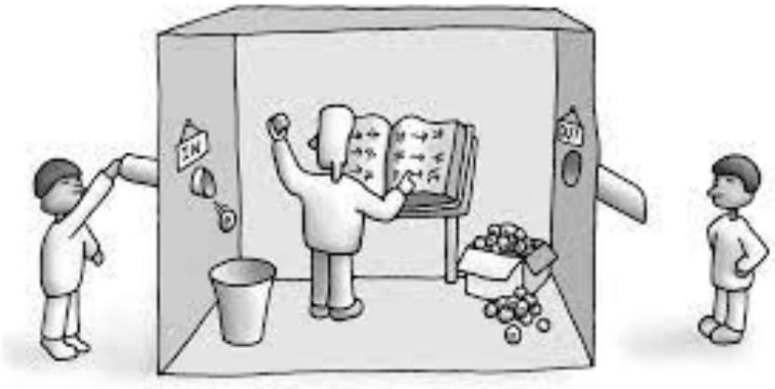


Pruebas de caja negra

- Consisten en diseñar una serie de **casos de prueba donde la entrada** de información conocida **permite evaluar los resultados de la salida** de información.
- Existen diferentes **técnicas**:
  - **Particiones de equivalencia**: se dividen los datos de entrada en el mínimo número de casos de uso posible (ejemplo posterior).
  - **Análisis de valores límite**: igual que la técnica anterior pero sólo aplicable para valores numéricos.
  - Etc.

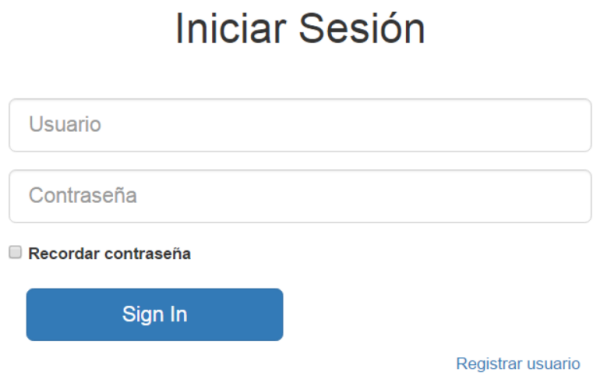
Pruebas de caja negra

- Este enfoque es más habitual, aunque no exclusivo, de las siguientes pruebas funcionales:
  - Pruebas de integración, en algunos casos.
  - Pruebas de **sistema**.
  - Pruebas de **aceptación**.



Pruebas de caja negra

- **Ejemplo**:
  - **Planteamiento**: Imaginemos un sistema que solo permite el acceso mediante validación de credenciales de usuario. Si una persona introduce un usuario y contraseña adecuados el sistema permite el acceso y, en caso contrario, lanza un mensaje: “Acceso denegado, usuario y contraseña no válidos“.

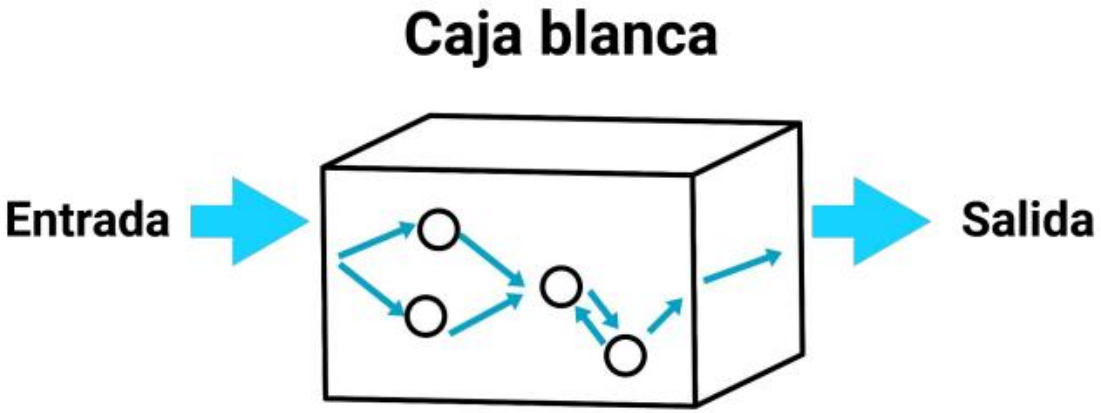


Pruebas de caja negra

- **Ejemplo**:
  - **Casos de prueba:**
    - Si introducimos un usuario incorrecto ➔ acceso denegado
    - Si introducimos un usuario válido y una contraseña incorrecta ➔ acceso denegado
    - Si introducimos un usuario válido y una contraseña válida ➔ acceso permitido

Pruebas de caja blanca

- Se **accede al detalle del código fuente** para llevarlas a cabo.
- Además de tener en cuenta las entradas y salidas de información.



Pruebas de caja blanca

- Consisten en diseñar una serie de **casos de prueba que permitan**:
  - Localizar **errores lógicos** en el código.
  - Comprobar que **todas las instrucciones** de un código **se llegan a ejecutar** al menos una vez. Especialmente en las estructuras **condicionales**.
  - Confirmar la correcta funcionalidad de los **bucles y sus límites**.
  - Acotar problemas de **rendimiento**.
  - Etc.

Pruebas de caja blanca

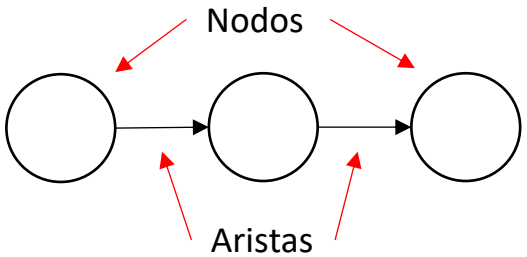
- La **técnica** más relevante es la **Prueba del camino básico**:
  - Se utiliza para diseñar una **serie de casos de prueba que garanticen la cobertura de todas las rutas** o caminos posibles en la ejecución de un código.

Pruebas de caja blanca. Técnica del camino básico

- **Pasos** para implementar la técnica de la **Prueba del camino básico**:

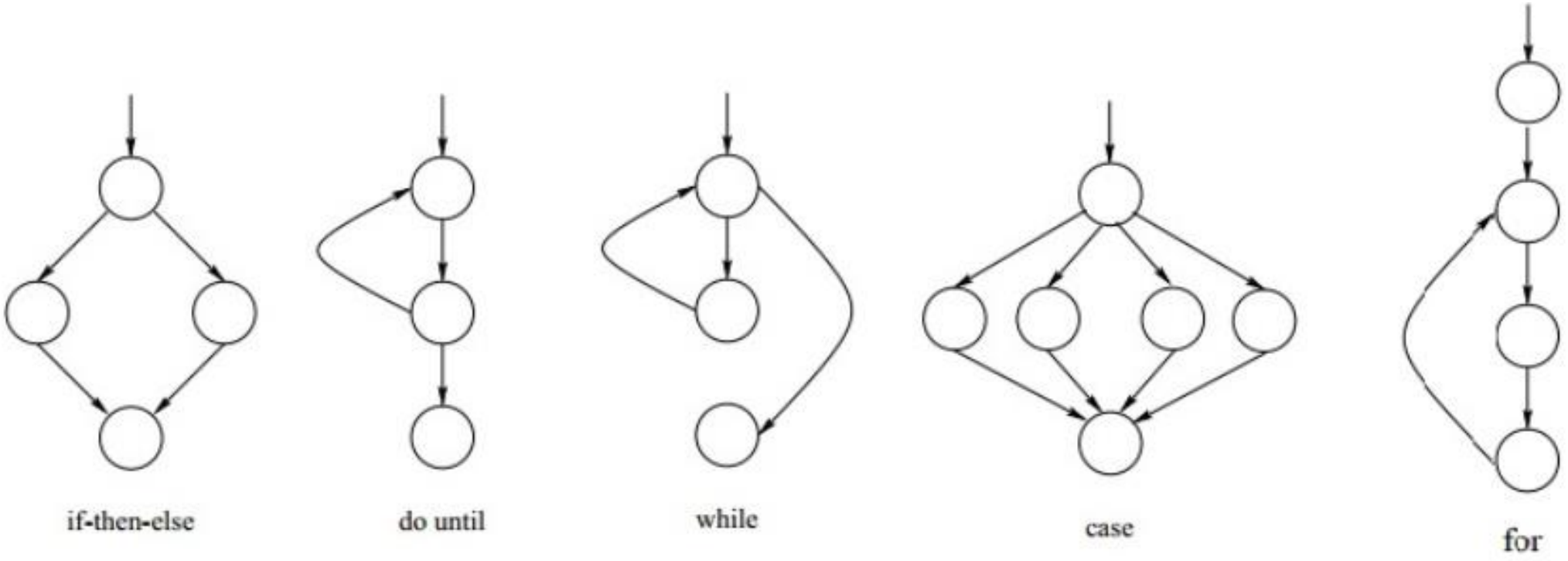
  1. **Diseñar el grafo de flujo de control (CFG) :**

     - Es una representación gráfica de los posibles flujos de ejecución de un código.
     - Cada grafo está formado por una serie de **nodos**, que representan las instrucciones o bloques de código.
     - Los nodos se conectan entre sí mediante las **aristas** (o arcos), que representan el flujo de control entre las instrucciones.



Pruebas de caja blanca. Técnica del camino básico

- **Estructuras de control en forma de grafo:**



Pruebas de caja blanca. Técnica del camino básico

2. Determinar la Complejidad Ciclomática:

- Cada **ruta o camino es una secuencia de nodos y aristas en el grafo** de flujo de control.
- Una **ruta o camino** se considera **independiente** si introduce **al menos una nueva arista** no cubierta por otros caminos.

Pruebas de caja blanca. Técnica del camino básico

2. **Determinar la Complejidad Ciclomática:**

   - La complejidad ciclomática es **una métrica que indica el número mínimo de caminos linealmente independientes** en un CFG.
   - Se calcula utilizando la **fórmula**:

$$V(G) = E - N + 2P$$

***Donde E es el número de aristas, N es el número de nodos, y P es el número de componentes conectados (generalmente $P = 1$).*

Pruebas de caja blanca. Técnica del camino básico

3. **Identificar Caminos Independientes**:
   - Identificar los caminos independientes basados en la complejidad ciclomática.
   - Cada camino debe introducir al menos una nueva arista no cubierta por otros caminos.

4. **Diseñar Casos de Prueba**:
   - Crear casos de prueba para ejecutar cada uno de los caminos independientes identificados.
   - Asegurarse de que los datos de entrada y las condiciones iniciales permiten la ejecución de cada camino.

Pruebas de caja blanca. Técnica del camino básico

- **Ejemplo:** dado un fragmento de código, utiliza la técnica del camino básico para diseñar los posibles casos de prueba. Las especificaciones del código son las siguientes :
  - Obtener 2 números aleatorios entre 0 y 1. Se almacenarán en las variables “a” y “b”.
  - Diseñar un bucle que itere “a+b” veces. En cada iteración:
    - Si alguno de los números aleatorios es 1, se sumará 5 a una variable “x”.
    - En caso contrario, se sumará 2 a una variable “x”.

Pruebas de caja blanca. Técnica del camino básico

- **Ejemplo:**

```
var a = Math.floor(Math.random()*2);                    Nodo 1(Inicializaciones)
var b = Math.floor(Math.random()*2);
var x = 0;

for(var i=0 ; i<a+b ; i++){                             Nodos 1(i=0),2(i<a+b),7(i++)
    if (a || b) {                                       Nodos 3(a==1),4(b==1)
        x = x + 5;                                      Nodo 5
    } else {
        x = x + 2;                                      Nodo 6
    }
}

console.log('a=' + a + '  b=' + b + '  x=' + x);       Nodo 8
```

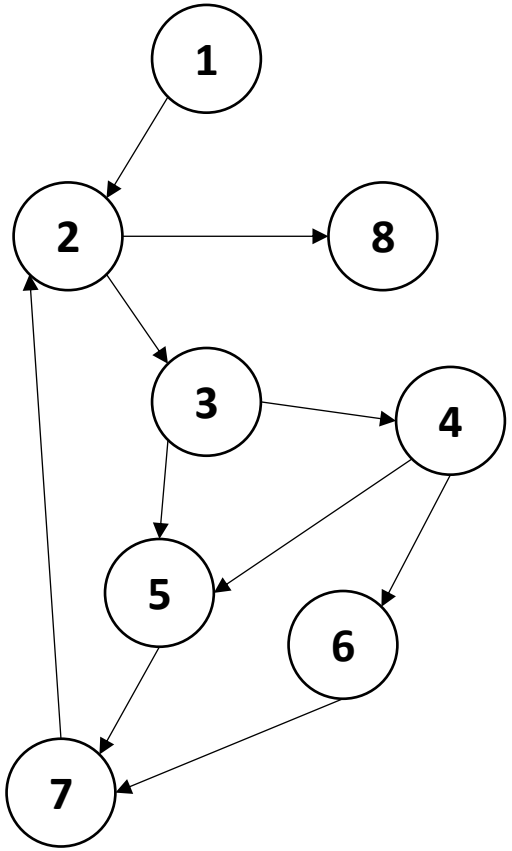
Pruebas de caja blanca. Técnica del camino básico

- **Ejemplo:**

```
var a = Math.floor(Math.random()*2);
var b = Math.floor(Math.random()*2);
var x = 0;

for(var i=0 ; i<a+b ; i++) {
    if (a || b) {
        x = x + 5;
    } else {
        x = x + 2;
    }
}

console.log('a=' + a + '  b=' + b + '  x=' + x);
```



Pruebas de caja blanca. Técnica del camino básico

- **Ejemplo:** aplicando la fórmula para determinar la complejidad ciclomática, obtenemos el número de caminos independientes.

$$V(G) = E - N + 2P$$

- V(G) = 10 aristas - 8 nodos + 2 = **4 caminos linealmente independientes**
  - Camino 1: 1, 2, 8.
  - Camino 2: 1, 2, 3, 5, 7, 2, 8.
  - Camino 3: 1, 2, 3, 4, 5, 7, 2, 8.
  - Camino 4: 1, 2, 3, 4, 6, 7, 2, 8.

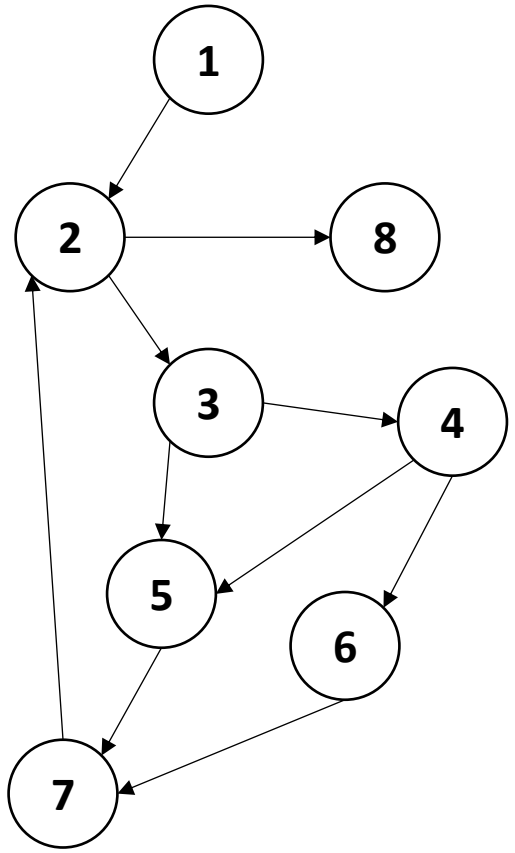
Pruebas de caja blanca. Técnica del camino básico

- **Ejemplo:** por lo tanto, la técnica del camino básico nos indica que para probar el fragmento de código de forma completa debemos diseñar, al menos, **4 casos de prueba**, basados en una precondición o situación conocida de entrada de información (“a” y “b”) y una precondición o resultado esperado:

  - **Caso de prueba 1:**
    - Precondición: a = 0 y b = 0
    - Postcondición: x = 2

Pruebas de caja blanca. Técnica del camino básico

- **Caso de prueba 2:**
  - Precondición: a = 1 y b = 0
  - Postcondición: x = 5
- **Caso de prueba 3:**
  - Precondición: a = 0 y b = 1
  - Postcondición: x = 5
- **Caso de prueba 4:**
  - Precondición: a = 1 y b = 1
  - Postcondición: x = 10



Pruebas de caja blanca. Técnica del camino básico

- **Ejemplo:** por último, se llevan a cabo las pruebas y los resultados se confrontan con lo esperado:
  - **Prueba 1:**
    - Resultado: x = 0
  - **Prueba 2:**
    - Resultado: x = 5
  - **Prueba 3:**
    - Resultado : x = 5
  - **Prueba 4:**
    - Resultado : x = 10

Pruebas de caja blanca. Técnica del camino básico

- **Ejemplo:** a modo de **informe final**, al confrontar los resultados con lo esperado (postcondición), se observa lo siguiente:
  - **Las pruebas correspondientes a los casos de prueba 2, 3 y 4 han sido satisfactorias.**

  - **La prueba correspondiente al caso de prueba 1 ha sido fallida:**
    - **Postcondición: x = 2**
    - **Resultado: x = 0**

Pruebas de caja blanca. Técnica del camino básico

- **Ejemplo:** como **conclusión,** con que uno de los casos de prueba sea fallido, el resultado general de las pruebas se considera fallido.
- Analizando el caso de prueba fallido, se puede concluir que la codificación no es correcta, ya que **una de las instrucciones es inaccesible**.

Pruebas de caja blanca. Técnica del camino básico

- **Ejemplo:** una posible corrección del código sería la siguiente:

```
var a = Math.floor(Math.random()*2);
var b = Math.floor(Math.random()*2);
var x = 0;

if (!a && !b) { // sería equivalente →  (!(a || b))
     x = x + 2;
} else {
    for(var i=0 ; i<a+b ; i++) {
        x = x + 5;
    }
}

console.log('a=' + a + '  b=' + b + '  x=' + x);
```

Pruebas de caja blanca. Técnica del camino básico

- **Ejemplo:** el concepto de **regresión** está directamente relacionado con la corrección que acabamos de aplicar.
- Cualquier corrección o modificación puede acarrear **efectos colaterales** en una solución software.
- Es interesante **volver a ejecutar las pruebas** de cualquier código que pueda verse afectado por una modificación.

Pruebas de caja blanca. Técnica del camino básico

- **Clasificación** general de la complejidad ciclomática:

Complejidad Ciclomática	Evaluación del Riesgo
1-10	Código sencillo, sin mucho riesgo
11-20	Relativa complejidad, riesgo moderado
21-50	Código complejo, alto riesgo
>50	Código no testeable, muy alto riesgo

Pruebas de caja blanca.

- **Resumen**. Las pruebas de caja blanca ofrecen las siguientes **ventajas**:
  - Permiten revisar en detalle **condiciones y bucles**.
  - Reducen la posibilidad **de errores no detectados.**
  - Facilitan la detección de **errores lógicos**.
  - Facilitan la **identificación temprana** de errores.
  - Ayudan a **identificar código redundante**, no utilizado o ineficiente.
  - Mejoran la comprensión del código y su **refactorización**.
  - Garantizan que todas las instrucciones en el código sean **ejecutadas al menos una vez**.

Pruebas de caja blanca.

- **Herramientas**. En un entorno de desarrollo existen una serie de herramientas que facilitan las tareas de generación de código satisfactorio de cara a superar pruebas:
  - El **depurador:** permite recorrer e inspeccionar un código instrucción a instrucción.
  - El **analizador de código:** muestra errores de compilación, advertencias y mensajes.
  - Los **gestores de pruebas unitarias:** automatizan las pruebas.

Pruebas de caja blanca

- Este enfoque es más habitual, aunque no exclusivo, de las siguientes pruebas funcionales:
  - Pruebas de integración, en algunos casos.
  - **Pruebas unitarias**.