PRUEBAS DE CAJA BLANCA.

TÉCNICA DEL CÁMINO BÁSICO

Este método fue diseñado por McCabe (1976).

Permite al diseñador de casos de prueba obtener **una medida de la complejidad lógica** del diseño procedimental, que se usará como guía para **definir un conjunto básico de caminos** de ejecución que recorrerán cada arista del programa al menos un vez.

Para ejecutar cada camino **se fija un caso de prueba**.

Los casos de prueba derivados con el método de McCabe **garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa**.

NOTACIÓN UTILIZADA: GRAFO DE FLUJO

Existen tres tipos de elementos:

- Nodos
- Aristas.
- Regiones.

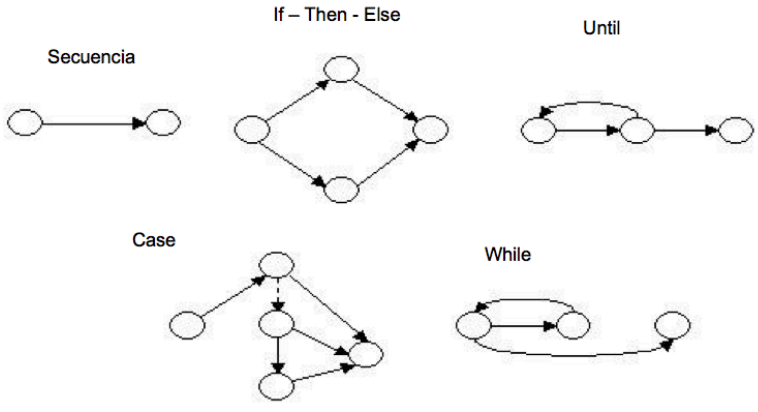
Cada círculo es un **nodo** y puede representar una o varias sentencias. Las flechas se llaman **aristas** y representan el flujo de control. Siempre una arista debe acabar en un nodo.

Grafo de flujo se denomina al grafo que se obtiene al representar las sentencias de un programa o módulo mediante la notación indicada.

¿CÓMO CONSTRUIR UN DIAGRAMA DE FLUJO A PARTIR DEL CÓDIGO?

1. Señalar todas las decisiones de las sentencias if-then-else, case-of, while y until
2. Agrupar resto de sentencias
3. Señalar los end if y end while

Las construcciones estructuradas se representan:



COMPLEJIDAD CICLOMÁTICA

La **complejidad ciclomática** de un grafo es una medida que nos proporciona la

información necesaria para conocer cuantos casos de prueba serán necesarios diseñar para que queden comprobados todos y cada uno de los flujos de control y sentencias del programa.

La complejidad ciclomática es una **métrica útil** para predecir los módulos que son más propensos a error. Puede ser usada tanto para planificar pruebas como para diseñar casos de prueba.

Por tanto esta medida **proporciona el número de caminos independientes** en el grafo de flujo asociado al programa, lo cual nos indica el **límite superior para el número de casos de prueba** que hay que realizar para asegurar que se ejecute al menos una vez cada sentencia, considerando como camino independiente cualquier camino del procedimiento o programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición.

Un camino independiente al resto de caminos ya probados es el que se mueve por una arista del grafo de flujo que no haya sido recorrida por los caminos anteriores.

Por tanto, la complejidad ciclomática se puede definir o calcular de la siguiente manera:

1) **V(G) = R**

   donde R es el número de regiones del grafo de flujo.

2) **V(G) = E – N + 2**

   donde E es el número de aristas del grafo de flujo y N es el número de nodos del grafo.

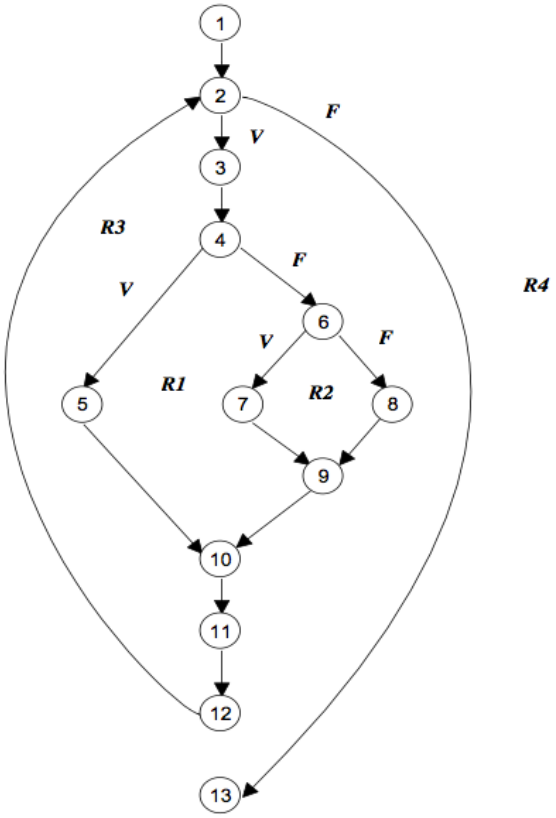
3) **V(G) = P + 1**

   donde P es el número de nodos predicados contenidos en el grafo de flujo G.

Un **nodo predicado** es aquel que posee una condición y se caracteriza porque de él salen dos o más aristas.

EJEMPLO:

```
    procedimiento buscar_elemento (vector1 ,vector2, elemento, longitud)
    inicio
1:  contador = 1
2:  mientras (contador <= longitud)
3:  hacer
    mostrar_pantalla("Procesando posición: ",  contador)
4:          si vector1[contador] = elemento
5:          entonces
                mostrar_pantalla("Coincidencia Encontrada posición:", contador)
6:          sino   si vector2[contador] = elemento
7:                 entonces
                        mostrar_pantalla("Coincidencia Encontrada posición:",
                                                                contador)
8:                 sino
                      vector2[contador] = 0
9:                 finsi
10:         finsi
11:         contador = contador + 1
12: finmientras
13:  fin
```



El cálculo de la complejidad ciclomática se puede llevar a cabo a través de las tres formas indicadas anteriormente:

1) El número de regiones del grafo de flujo. Como se indica en el grafo de flujo el número de regiones cerradas son tres. Además se suma una región adicional que se corresponde con una región genérica donde se incluye todo el grafo de flujo.

V(G) = 4.

2) La complejidad ciclomática V(G) de un grafo de flujo G se define como:

V(G) = E – N + 2 =15 – 13 + 2 = 4

Número de aristas: E = 15

Número de nodos: N = 13

3) La complejidad ciclomática V(G), de un grafo de flujo G también se define como.

V(G) = P + 1 =3 +1= 4

donde P es el número de nodos predicados contenidos en el grafo de flujo G. Un nodo predicado es aquel que posee una condición y se caracteriza por que de él salen dos o más aristas.

Como se puede observar, sea cual sea el método elegido para el cálculo de la complejidad ciclomática el resultado obtenido es siempre el mismo.

El hecho de que la complejidad ciclomática V(G) sea 4 indica que con cuatro casos de prueba que recorran los cuatro caminos linealmente independientes se podrá comprobar toda la lógica interna del procedimiento.

Los caminos independientes que se obtengan a partir del grafo no tienen por que ser siempre los mismos, lo que sí es necesario es que **en cada camino se incluya una nueva arista no visitada hasta entonces.**

Un conjunto de caminos independientes sería:

- **Camino 1**: 1-2-13 Aristas Nuevas Recorridas: 1-2, 2-13
- **Camino 2**: 1-**2-3-4-5-10-11-12-2**-13 Aristas Nuevas Recorridas: 2-3, 3-4, 4-5, 5-10, 10-11, 11-12, 12-2
- **Camino 3**: 1-2-3-**4-6-7-9-10**-11-12-2-13 Aristas Nuevas Recorridas: 4-6, 6-7, 7-9, 9-10
- **Camino 4**: 1-2-3-4-**6-8-9**-10-11-12-2-13 Aristas Nuevas Recorridas: 6-8, 8-9.

PASOS PAR EL DISEÑO DE CASOS DE PRUEBA

1. Dibujar el grafo de flujo.
2. Calcular la complejidad ciclomática.
3. Determinar el conjunto de caminos independientes.
4. Preparar el caso de prueba para cada camino.

EJEMPLO:

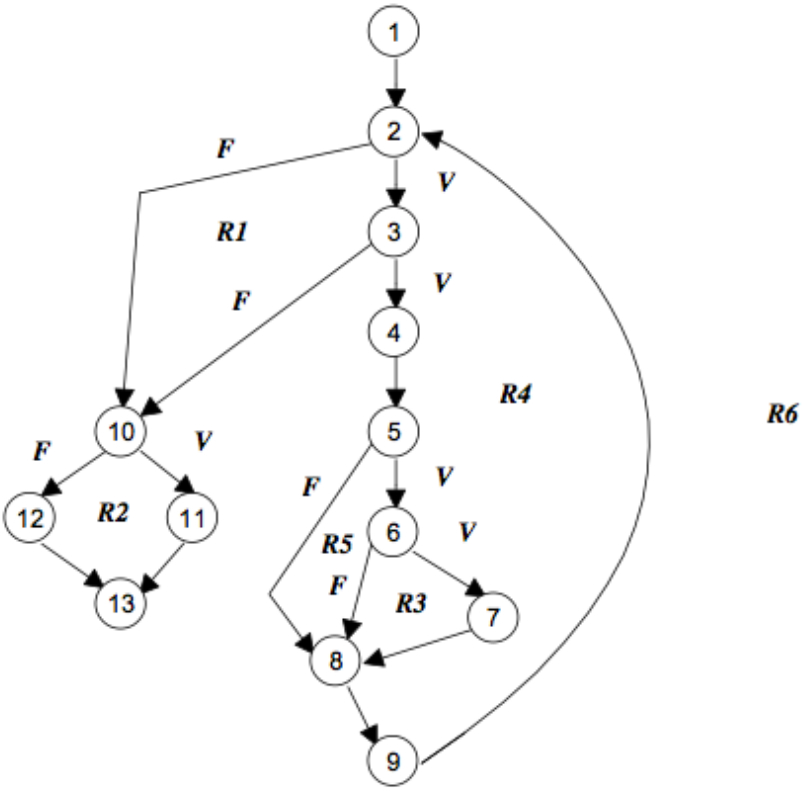
Procedimiento que calcula la media de hasta 100 números que se encuentren entre límites. También calcula el total de entradas y el total de válidos.

```
PROCEDURE media;
    * Este procedimiento calcula la media de 100 o menos
      números que se encuentren entre unos limites;
      también calcula el total de entradas y el total
      de números válidos.

      INFERFACE RETURNS media, total. entrada, total. válido;
      INTERFACEACEPTS valor, mínimo, máximo;

      TYPE valor [1:100] IS SCALAR ARRAY;
      TYPE media, total. entrada, total. válido;
         Mínimo, máximo, suma IS SCALAR:
      TYPE i IS INTEGER;
      i = 1;
      total, entrada =total, válido = 0;
      suma =0;
      DO WHILE VALOR [i] <> –999 and total entrada < 100
         Incrementar total entrada en 1;
         IF valor [i] >= mínimo  AND  valor [i] <= maximo
            THEN incrementar total.válido en 1;
                 suma = suma +valor [i];
            ELSE ignorar
         ENDIF
         Incrementar i en 1;
      ENODO
      If total valido > 0
         THEN media = suma/total valido,
         ELSE media = –999,
      ENDIF
END media
```

PASO 1:



PASO 2:

Calcular la complejidad ciclomática:

V(G) = 6 regiones.

V(G) = 17 aristas – 13 nodos + 2 = 6

V(G) = nodos predicado + 1 = 5 + 1 = 6

PASO 3:

Determinar los 6 caminos independientes:

Camino 1: 1-2-10-11-13

Camino 2: 1-2-**10-12-13**

Camino 3: 1-**2-3-10**-11-13

Camino 4: 1-2-**3-4-5-8-9-2**-…

Camino 5: 1-2-3-4-**5-6-8**-9-2-…

Camino 6: 1-2-3-4-5-**6-7-8**-9-2- …

Los puntos suspensivos que siguen a los caminos 4,5 y 6 indican que cualquier camino del resto de la estructura de control es aceptable.

PASO 4:

- **Caso de prueba del camino 1: Camino 1: 1-2-10-11-13**

Para el camino 1, se coge un valor[k] tenga una entrada válida donde k < i, y también se tiene en cuenta que valor[i]=-999. Para que se de este camino se tienen que dar estos valores, lo cual sólo se puede hacer si este camino está incluido en otro camino mayor que permita hacer total_valido > 0, ya que en caso contrario no podría ocurrir. Los resultados esperados son una media correcta y entradas válidas.

  - Valor[k] = entrada válida, con k<i, definida abajo.
  - Valor[i] = -999, donde 2 <= i <= 100
  - Resultados esperados: Media correcta sobre los n valores y totales adecuados
  - Nota: no se puede probar por sí sola: debe ser probada como parte de los caminos 4,5 y 6.

- **Caso de prueba del camino 2: Camino 2: 1-2-10-12-13**

Para el camino 2 es igual que el anterior, pero en este caso existe un valor, el primero, igual a - 999, es decir, no se entra en el bucle mientras. Se recorre el grafo por el nodo. Este camino se puede recorrer de forma independiente, no hace falta recorrer los otros caminos.

  - valor[1] = -999
  - Resultados esperados: Media = -999; otros totales con sus valores iniciales.

- **Caso de prueba del camino 3 Camino 3: 1-2-3-10-11-13**

Para probar el camino 3 vemos que no se cumple la segunda condición del mientras. Se probará intentando procesar un número de entradas mayores de 100 (101 o más valores). Los primeros 100 valores deben de ser válidos y los resultados esperados son los mismos que en el camino 1, donde se ejecuta mientras para 100 entradas y después sale de él.

  - Intento de proceder 101 o más valores.
  - Los primeros 100 valores deben ser válidos.
  - Resultados Esperados: Media correcta sobre los n valores y totales adecuados

- **Caso de prueba del camino 4: Camino 4: 1-2-3-4-5-8-9-2- .... -> cualquier camino elegido ya es válido como por ejemplo el 2-10-11-13**

  Para el camino 4, en el **si** dentro del **mientras** existe una salida. Valor[i] es menor que mínimo. El diseño de este caso de prueba tiene como objetivo comprobar un valor de entrada menor que el mínimo establecido. Resultado: media correcta entre límites mínimo y máximo. Sale del **si** cuando valor[k] < mínimo.

    - valor[i] = entrada valida con i < 100
    - valor[k] < minimo para k<i

• Resultados Esperados: Media correcta sobre los valores y totales adecuados.

- **Caso de prueba del camino 5: Camino 5: 1-2-3-4-5-6-8-9-2- ...**
  Para el camino 5, hay que comprobar un valor de k mayor que el valor de máximo establecido. Resultado: media correcta entre mínimo y máximo y número de entradas.

  • valor[i] = entrada valida con i <100

  • valor[k] > maximo para k<=i

  • Resultados Esperados: Media correcta sobre los n valores y totales adecuados.

- **Caso de prueba del camino 6: Camino 6: 1-2-3-4-5-6-7-8-9-2 ...**

  Para el camino 6 existen n valores que cumplen las condiciones del procedimiento. Se obtienen medias y entradas adecuadas. Es el caso más general.

  • valor[i] = entrada válida con i < 100

  • Resultados Esperados: Media correcta sobre los n valores y totales adecuados.