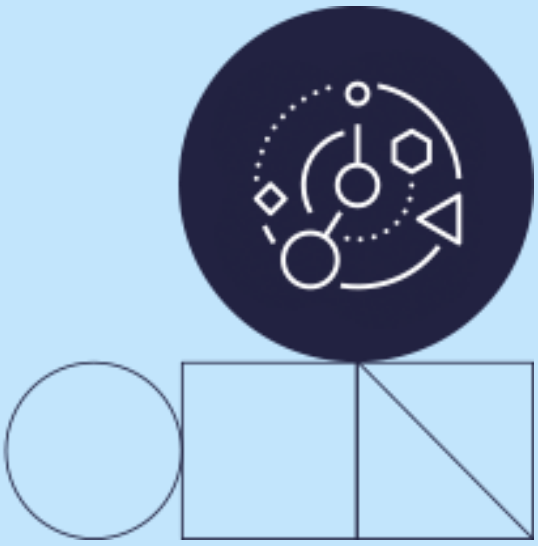
IBM SkillsBuild

IBM

Conceptos básicos de programación

Estructuras de control: condicionales y bucles



Índice

Introducción

Son parte fundamental de cualquier lenguaje.

Sin ellas, las instrucciones de un programa sólo podrían ejecutarse en el orden en que están escritas (orden secuencial).

Las estructuras de control permiten modificar este orden.

Hay dos categorías de estructuras de control:

- Condicionales
- Bucles

Condicionales

Permiten que se ejecuten conjuntos distintos de instrucciones, en función de que se verifique o no determinada condición.

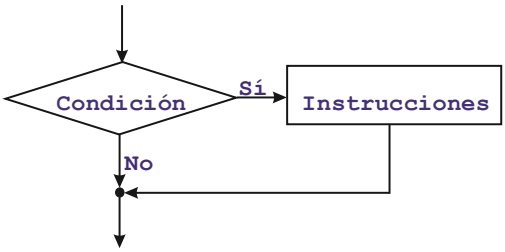
En términos de un lenguaje de programación, que se verifique o no una condición se traduce en que una (adecuada) expresión lógica tome el valor verdadero (*TRUE*) o tome el valor falso (*FALSE*).

En los casos más sencillos y habituales la condición suele ser una comparación entre dos datos, como por ejemplo: si **a < b** hacer una cosa y en caso contrario hacer otra distinta.

Estructura condicional simple: IF

Este es el tipo más sencillo de estructura condicional. Sirve para implementar acciones condicionales del tipo siguiente:

- Si se verifica una determinada condición, ejecutar una serie de instrucciones y luego seguir adelante.
- Si la condición NO se cumple, NO se ejecutan dichas instrucciones.



...

if condición

instrucciones a cumplir en caso afirmativo

...

Obsérvese que, en ambos casos (que se verifique o no la condición), los “caminos” bifurcados se unen posteriormente en un punto, es decir, el flujo del programa recupera su carácter secuencial, y se continúa ejecutando por la instrucción siguiente a la estructura IF.

Como ejemplo de utilización de este tipo de condicional, se considera el cálculo del valor en un punto *x* de una función definida por partes, como por ejemplo:

$$f(x) = \begin{cases} 0 & \text{si } x \leq 0 \\ x^2 & \text{si } x > 0 \end{cases}$$

A continuación, se muestra el pseudocódigo correspondiente.

Cálculo del valor de la función $f(x) = 0$ si $x \leq 0$, $f(x) = x^2$ si $x > 0$.

Inicio

1- LEER x
2- HACER f=0 3- **Si** x>0 HACER f=x^2
Fin Si

4- IMPRIMIR 'El valor de la función es: ', f

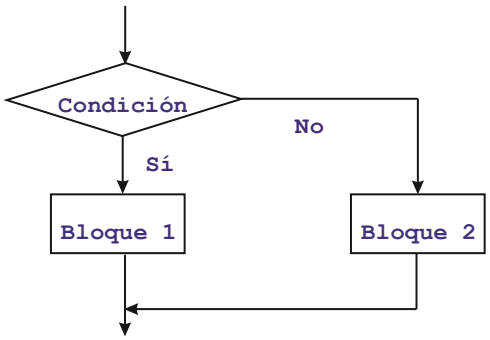
Fin

Estructura condicional doble: IF – ELSE

Este tipo de estructura permite implementar condicionales en los que hay dos acciones alternativas:

- Si se verifica una determinada condición, ejecutar un serie de instrucciones (bloque 1).
- Si no, esto es, si la condición NO se verifica, ejecutar otra serie de instrucciones (bloque 2).

En otras palabras, en este tipo de estructuras hay una alternativa: se hace una cosa o se hace la otra. En ambos casos, se sigue por la instrucción siguiente a la estructura **IF** - **ELSE**.



...

if

condición

bloque-1

else

bloque-2

...

Como ejemplo de utilización de este tipo de estructuras se plantea el problema de calcular las raíces de una ecuación de segundo grado

$ax^2 + bx + c = 0$

distinguiendo dos casos: que las raíces sean reales o que sean complejas (no se contempla, de momento, dintinguir entre una o dos raíces reales).

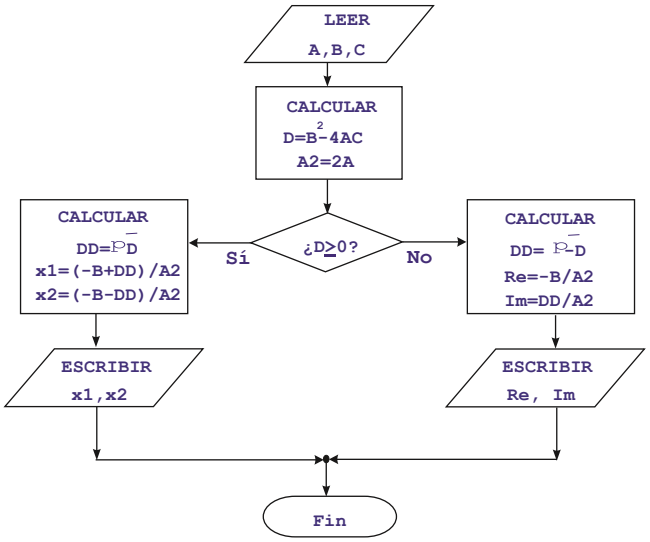


Diagrama de flujo para determinar las raíces reales o complejas de la ecuación de segundo grado $Ax^2 + Bx + C = 0$.

A continuación, se muestra el pseudocódigo correspondiente.

Cálculo de las raíces de la ecuación de segundo grado $Ax^2 + Bx + C = 0$, distinguiendo los casos de raíces reales y complejas.

Inicio

1- LEER A,B y C
2- CALCULAR D=B^2-4*A*C
3- CALCULAR AA=2*A
4- **Si** D≥0 √

CALCULAR DD= D x1=(-B+DD)/AA x2=(-B-DD)/AA

IMPRIMIR 'La ecuación tiene raíces reales:', x1, x2

Si no √

CALCULAR DD= -D

Re=-B/AA

Im=DD/A2

IMPRIMIR 'La ecuación tiene raíces complejas conjugadas:'

IMPRIMIR 'Parte real:', Re

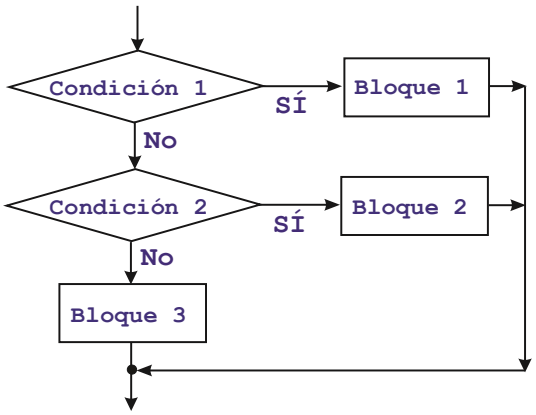
IMPRIMIR 'Parte imaginaria:', Im **Fin Si** Fin

Estructura condicional múltiple: IF - ELSEIF – ELSE

En su forma más general, la estructura IF - ELSEIF - ELSE permite implementar condicionales más complicados, en los que se "encadenan" condiciones en la forma siguiente:

- Si se verifica la condición 1, ejecutar las instrucciones del bloque 1.
- Si no se verifica la condición 1, pero SÍ se verifica la condición 2 , ejecutar las instrucciones del bloque 2.
- Si no, esto es, si no se ha verificado ninguna de las condiciones anteriores, ejecutar las instrucciones del bloque 3.

En cualquiera de los casos, el flujo del programa continúa por la instrucción siguiente a la estructura



...

if condici ´on-1

bloque-1

elseif condici ´on-2

bloque-2

else bloque-3

...

A continuación, se muestra el pseudocódigo correspondiente.

Determinación del signo de un número: positivo, negativo o nulo.

Inicio

1- LEER X
2- **Si** X>0

IMPRIMIR 'El número tiene signo positivo'

Si no, si X<0

IMPRIMIR 'El número tiene signo negativo'

Si no

IMPRIMIR 'El número es nulo' Fin

La cantidad de casos que podemos agregar a este tipo de condicionales es infinita.

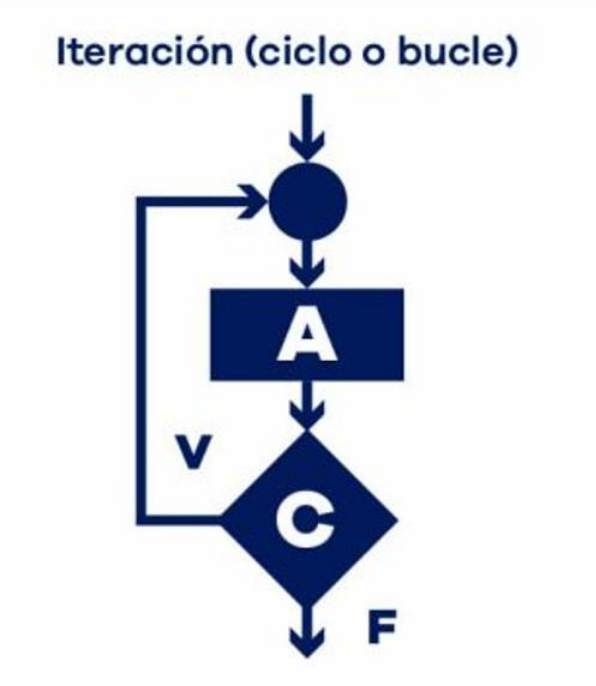
E incluso podemos hacer condicionales anidados unos dentro de otros, con lo que la complejidad sube exponencialmente.

Dados dos números reales, **a** y **b**, y el símbolo, **S** (carácter), de un operador aritmético (**+**, **-**, *****, **/**), imprimir el resultado de la operación **a S b**

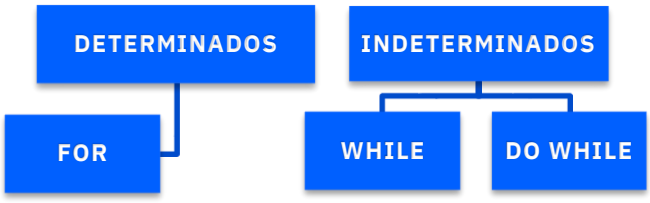
```
Inicio
    LEER a
    LEER b
    LEER S
    Si S='+'
        IMPRIMIR 'El resultado es =', a+b
    Si no, si S='-'
        IMPRIMIR 'El resultado es =', a-b
    Si no, si S='*'
        IMPRIMIR 'El resultado es =', a*b
    Si no, si b=0
        Si a=0
            IMPRIMIR 'El resultado es =', NaN
            (indeterminación)
        Si no
            IMPRIMIR 'El resultado es =', Inf (infinito)
        Fin Si
    Si no
        IMPRIMIR 'El resultado es =', a/b
Fin Si Fin
```

Bucles

Permiten que se ejecute repetidamente un conjunto de instrucciones, bien un número pre-determinado de veces, o bien hasta que se verifique una determinada condición.



Existen dos tipos de bucles según si se conoce o no el número de repeticiones que va a hacer nuestro bucle.



En los bucles determinados se conoce con certeza el número de veces que se va a repetir, por ejemplo podemos meter en un bucle una frase y que se nos muestre en pantalla un número determinado de veces.

En los bucles indeterminados no sabemos a ciencia cierta cuántas veces se va a repetir un bucle, ya que se estará repitiendo hasta que se cumpla una condición que, a priori, no sabemos cuándo se cumplirá.

Por ejemplo, imprimir repetidamente en pantalla una frase hasta que el usuario pulse una tecla para detener el bucle.

A continuación, se describen las distintas estructuras de control. Para cada una de ellas se describe el diagrama de flujo y una sintaxis genérica que será ligeramente diferente según el lenguaje que estemos usando, pero básicamente en todos los lenguajes de alto nivel es muy parecida.

Obsérvese que todas ellas tienen una única entrada y una única salida.

Estructura de repetición indexada: FOR

Este tipo de estructura permite implementar la repetición de un cierto conjunto de instrucciones un número pre-determinado de veces.

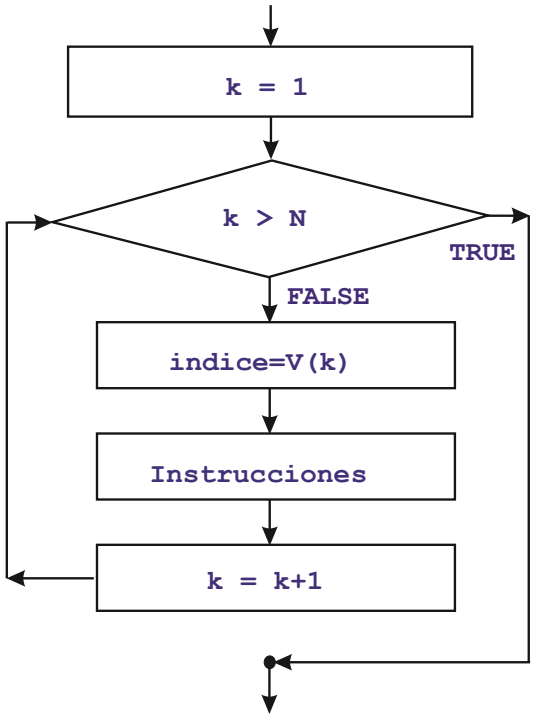
Para ello se utiliza una variable de control del bucle, llamada también índice, que va recorriendo un conjunto pre-fijado de valores en un orden determinado. Para cada valor del índice en dicho conjunto, se ejecuta una vez el mismo conjunto de instrucciones.

En el siguiente ejemplo, el bloque de instrucciones se ejecuta una vez para cada valor del índice, que va tomando sucesivamente el valor de cada componente del vector V, de longitud N.

... **for** indice=V

instrucciones

...



El indice del bucle recorre los valores de un vector V de longitud N.

Como ejemplo de utilización de la estructura FOR, véanse el siguiente pseudo-códigos para calcular la suma de los n primeros números impares.

- El valor de la variable de control índice puede ser utilizado o no dentro del conjunto de instrucciones que forman parte del cuerpo del FOR, pero no debe ser modificado.
- El conjunto de valores que debe recorrer el índice puede ser **vacío** (N=0). En ese caso, el bloque de instrucciones no se ejecuta ninguna vez.
- Las estructuras FOR e IF pueden "anidarse", es decir, incluir una dentro de la otra, con la restricción (de sentido común) de que la interior tiene que estar completamente contenida en uno de los bloques de instrucciones de la otra.

Dado un entero, n, calcular la suma de los n primeros números impares.

Inicio

LEER n

HACER suma=0

Para i= 1, 3, 5, ..., 2*n-1 HACER suma=suma+i

Fin Para

IMPRIMIR 'La suma vale : ', suma

Fin

Algoritmo 5.10 Dado un número natural, n, imprimir la lista de sus divisores, en orden decreciente.

Inicio LEER n

IMPRIMIR ' Lista de divisores del numero: ', n

Para i=ParteEntera(n/2) hasta 2 (incremento -1)

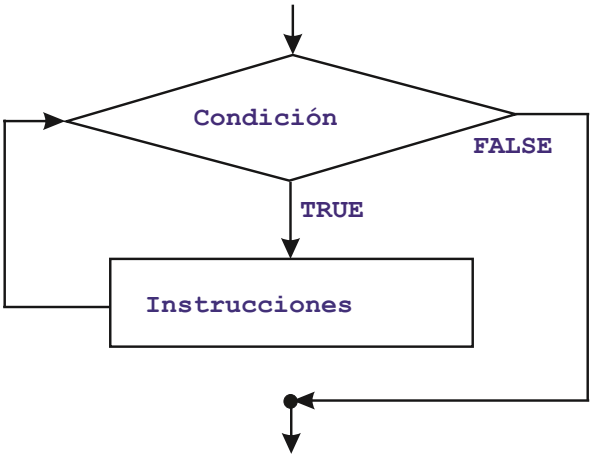
Si resto(n/i)=0 IMPRIMIR i

Fin Si Fin Para

IMPRIMIR 1 Fin

Estructura repetitiva condicional: WHILE

Permite implementar la repetición de un mismo conjunto de instrucciones mientras que se verifique una determinada condición: el número de veces que se repetirá el ciclo no está definido *a priori*. El diagrama de flujo descriptivo de esta estructura se muestra en la siguiente figura.



...

while expresión-lógica

instrucciones

end

...

Su funcionamiento es evidente, a la vista del diagrama:

- Al comienzo de cada iteración se evalúa la expresión-lógica.
- Si el resultado es VERDADERO, se ejecuta el conjunto de instrucciones y se vuelve a iterar, es decir, se repite el paso 1.
- Si el resultado es FALSO, se detiene la ejecución del ciclo WHILE y el programa se sigue ejecutando por la instrucción siguiente al END.

A continuación se representa es un ejemplo de utilización de esta estructura.

Imprimir de forma ascendente los 100 primeros números naturales.

Inicio i=1

Mientras que i ≤ 100

IMPRIMIR i

HACER i=i+1

Fin Mientras Fin

Ruptura de ciclos de repetición: BREAK y CONTINUE

En ocasiones es necesario interrumpir la ejecución de un ciclo de repetición **en algún punto interno del bloque de instrucciones que se repiten**.

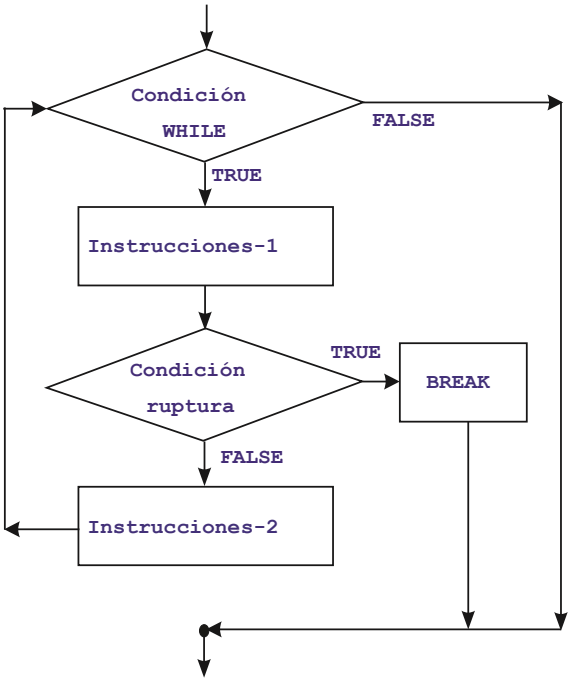
Lógicamente, ello dependerá de que se verifique o no alguna condición.

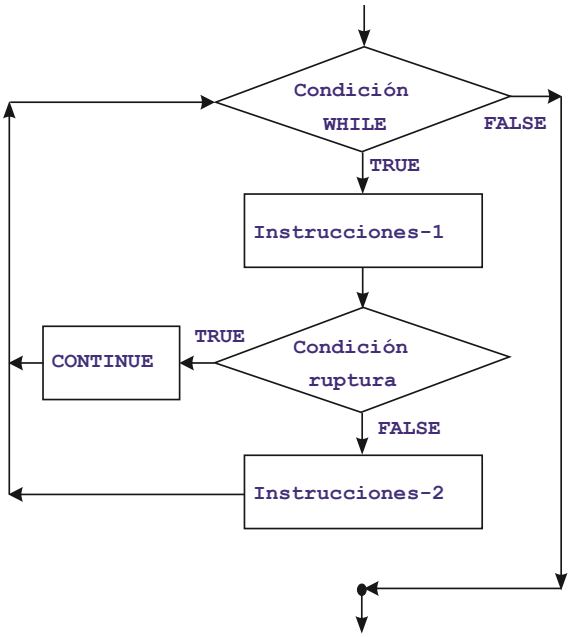
La interrupción puede hacerse de dos formas:

1. Abandonando el ciclo de repetición definitivamente.
2. Abandonando la iteración en curso, pero comenzando la siguiente.

Las instrucciones para poner esto en práctica tienen nombres diversos en los distintos lenguajes de programación. En Javascript, la primera opción se implementa con la instrucción BREAK y la segunda con la instrucción CONTINUE. Ambas pueden utilizarse tanto para romper un ciclo FOR como un ciclo WHILE. Cuando se utiliza la orden BREAK dentro de un ciclo FOR, el índice del bucle conserva, fuera del mismo, el último valor que tomó.

Véanse los correspondientes diagramas de flujo en las siguientes figuras.





Estructura de elección entre varios casos: SWITCH

Este tipo de estructura permite decidir entre varios caminos posibles, en funci´on del valor que tome una determinada instrucción.

No todos los lenguajes de programación disponen de este estructura.

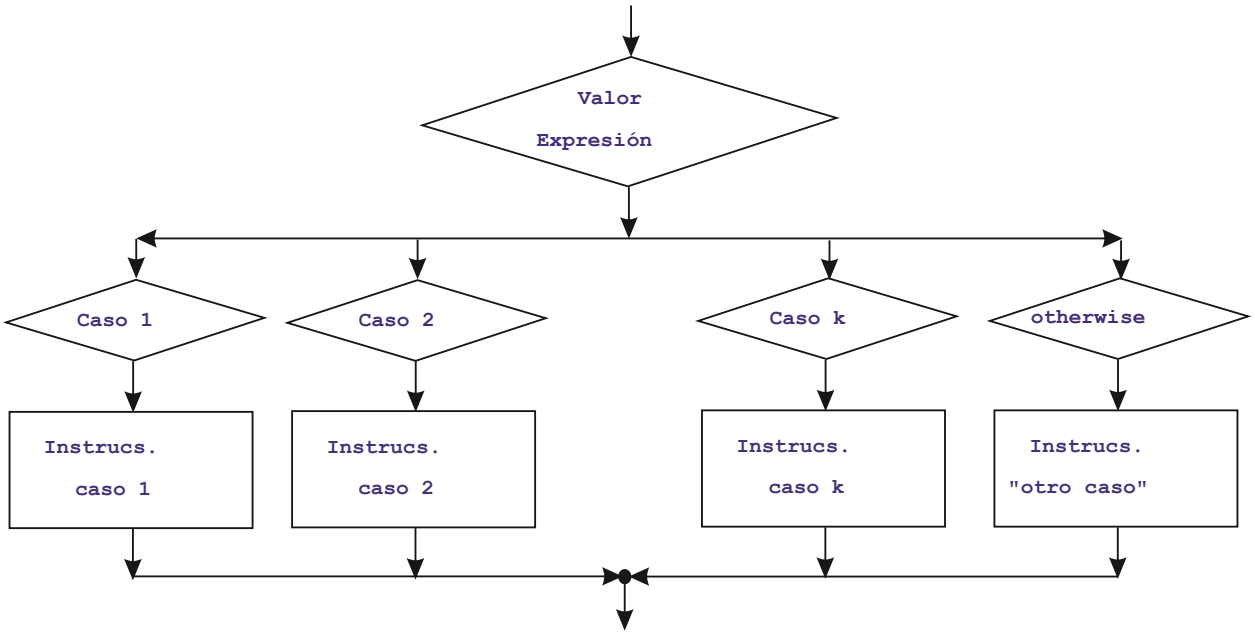
El diagrama de flujo correspondiente a una de estas estructuras (con cuatro casos) se presenta en la Figura 5.12.

switch expresión

case valor-1 instrucciones caso 1

case valor-2 instrucciones caso 2 ... **case** {valores...}

default instrucciones caso diferente
end



En cada uno de los casos, el valor correspondiente puede ser o bien un sólo valor, o bien un conjunto de valores, en cuyo caso se indican entre llaves. La cláusula DEFAULT y su correspondiente conjunto de instrucciones puede no estar presente.

El funcionamiento es el siguiente:

- Al comienzo se evalúa la expresión.
- Si expresión toma el valor (ó valores) especificados junto a la primera cláusula CASE, se ejecuta el conjunto de instrucciones de este caso y después se abandona la estructura SWITCH, continuando por la instrucción siguiente al END.
- Se repite el procedimiento anterior, de forma ordenada, para cada una de las cláusulas CASE que siguen.
- Si la cláusula DEFAULT está presente y la expresión no ha tomado ninguno de los valores anteriormente especificados, se ejecuta el conjunto de instrucciones correspondiente.

Obsérvese que se ejecuta, **como máximo** el conjunto de instrucciones de uno de los casos, es decir, una vez que se ha verificado un caso y se ha ejecutado su conjunto de instrucciones, no se testea el resto de casos, ya que se abandona la estructura. Obviamente, si la cláusula DEFAULT no está presente, puede ocurrir que no se dé ninguno de los casos.

Ejemplo de utilización de la sentencia SWITCH.

Dados dos números reales, **a** y **b**, y el símbolo, **S** (carácter), de un operador aritmético (**+**, **-**, *****, **/**), imprimir el resultado de la operación **a S b**

```
LEER a , b , S
Elegir caso S
  Caso '+'
    IMPRIMIR 'El resultado es =', a+b
  Caso '-'
    IMPRIMIR 'El resultado es =', a-b
  Caso '*'
    IMPRIMIR 'El resultado es =', a*b
  Caso '*'
    IMPRIMIR 'El resultado es =', a*b
  Caso '/' Si a6=0
      IMPRIMIR 'El resultado es =', a/b
    Si no, si b6=0
      IMPRIMIR 'El resultado es =', Inf (infinito)
    Si no
      IMPRIMIR 'El resultado es =', NaN (indeterminaci´on) Fin Si
  En otro caso
IMPRIMIR 'El operador no se reconoce' Fin
```

NOTA: Todas las estructuras de control (también llamados modificadores del flujo de ejecución) presentados en este tema son genéricos. La mayoría de los lenguajes de programación disponen de ellas, aunque no todos y su sintaxis difiere ligeramente de uno a otro.

Más adelante en este curso veremos las que corresponden a *Javascript* y a *Python*.