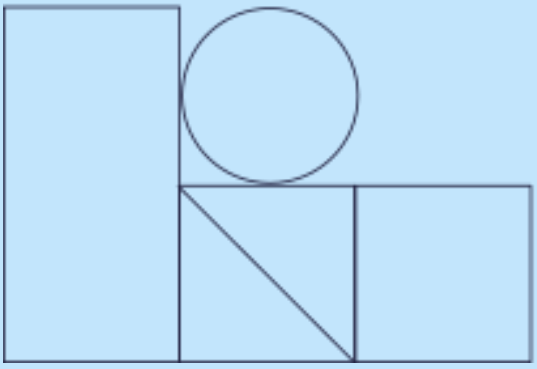
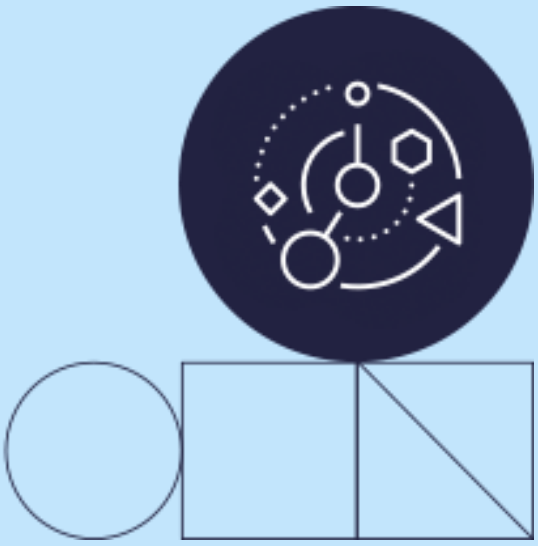
IBM SkillsBuild

IBM

Noções básicas de programação

Estruturas de controle: condicionadores e loops





Índice

Introducción

Eles são uma parte fundamental de qualquer idioma.

Sem elas, as instruções em um programa só poderiam ser executadas na ordem em que são escritas (ordem seqüencial).

As estruturas de controle permitem que esta ordem seja modificada.

Há duas categorias de estruturas de controle:

- Condicional
- Laços

Condicional

Eles permitem que diferentes conjuntos de instruções sejam executados, dependendo se uma determinada condição é verificada ou não.

Em termos de uma linguagem de programação, se uma condição é ou não verificada se traduz em uma expressão lógica (adequada) tomando o valor VERDADEIRO (*TRUE*) ou tomando o valor FALSO (*FALSE*).

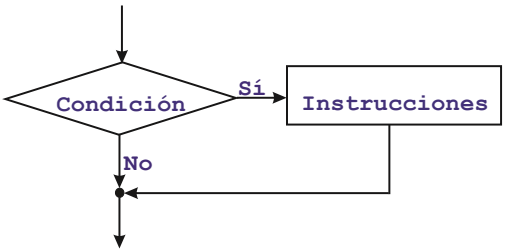
Nos casos mais simples e mais comuns, a condição é geralmente uma comparação entre dois dados, tais como: se **a < b** faz uma coisa e de outra forma faz outra coisa.

Estrutura condicional simple: IF

Este é o tipo mais simples de estrutura condicional. É utilizado para implementar ações condicionais do seguinte tipo:

- Se uma determinada condição for cumprida, executar uma série de instruções e, em seguida, proceder.

- Se a condição NÃO for cumprida, NÃO execute essas instruções.



...

if condición

instruções a serem seguidas em caso afirmativo

...

Observe que, em ambos os casos (quer a condição seja ou não verificada), os "caminhos" bifurcados são posteriormente unidos em um ponto, ou seja, o fluxo do programa recupera seu caráter seqüencial, e continua a ser executado pela instrução seguindo a estrutura IF.

Como exemplo do uso deste tipo de condicional, consideramos o cálculo do valor em um ponto x de uma função definida por partes, por exemplo:

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

O pseudocódigo correspondente é mostrado abaixo.

Cálculo do valor da função f(x) = 0 se x ≤ 0, f(x) = x2 se x > 0.

Início

1- LER x

2- MAKE f=0 3- **Se** x>0 MAKE f=x2

Fim Se

4- PRINT 'O valor da função é: ', f

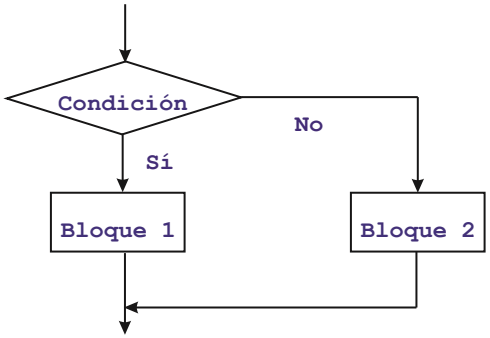
Fim

Estrutura condicional dupla: IF – ELSE

Este tipo de estrutura permite a implementação de condicionantes nos quais existem duas ações alternativas:

- Se uma determinada condição for verificada, executar uma série de instruções (bloco 1).
- Caso contrário, isto é, se a condição NÃO for cumprida, executar outro conjunto de instruções (bloco 2).

Em outras palavras, neste tipo de estrutura há uma escolha: você faz uma coisa ou faz a outra. Em ambos os casos, é seguido pela instrução após a estrutura **IF - ELSE**.



...

if

condición

bloco -1

else

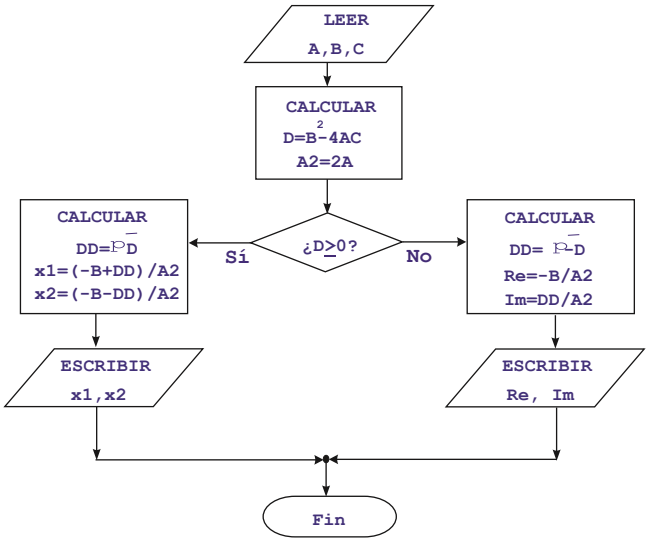
bloco -2

...

Um exemplo do uso deste tipo de estrutura é o problema de calcular as raízes de uma equação de segundo grau

$ax^2 + bx + c = 0$

distinguindo entre dois casos: que as raízes são raízes reais ou que são complexas (por enquanto, não está prevista a distinção entre uma ou duas raízes reais).



Fluxograma para determinar as raízes reais ou complexas da equação de segundo grau Ax2 + Bx + C = 0.

O pseudocódigo correspondente é mostrado abaixo.

Cálculo das raízes da equação de segundo grau Ax2 + Bx + C = 0, distinguindo os casos de raízes reais e complexas.

Início

1- LEIA A, B e C

2- CALCULAR D=B2-4*A*C

3- CALCULAR AA=2*A

4- Se D≥0 √

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

IMPRESSÃO 'A equação tem raízes reais:', x1, x2

Se não √

CALCULAR DD= -D

Re=-B/AA

Im=DDD/A2

IMPRESSÃO 'A equação tem raízes complexas conjugadas:', x1, x2

PRINT 'Real part:', Re

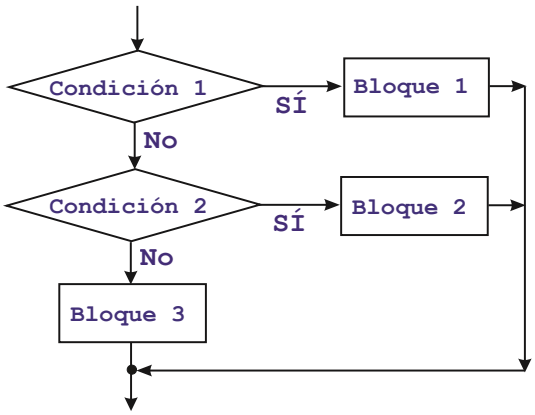
PRINT 'Parte imaginária:', Im End Sim End

Estrutura condicional múltipla: IF - ELSEIF – ELSE

Em sua forma mais geral, a estrutura IF - ELSEIF - ELSE permite a implementação de condições mais complicadas, nas quais as condições são "encadeadas" da seguinte maneira:

- Se a condição 1 for verificada, executar as instruções do bloco 1.
- Se a condição 1 não for verificada, mas a condição 2 for verificada, executar as instruções do bloco 2.
- Caso contrário, ou seja, se nenhuma das condições acima for verificada, executar as instruções do bloco 3.

Em ambos os casos, o fluxo do programa continua através da instrução seguindo a estrutura.



...

if

condición-1

bloque-1

elseif condici´on-2

bloque-2

else bloque-3

...

O pseudocódigo correspondente é mostrado abaixo.

Determinação do sinal de um número: positivo, negativo ou nulo.

Início

1- LEIA X

2- Se X>0

IMPRIMIR "O número tem um sinal positivo".

Caso contrário, se X<0

IMPRIMIR "O número tem um sinal negativo".

Caso contrário

IMPRIMIR "O número é nulo" Fim

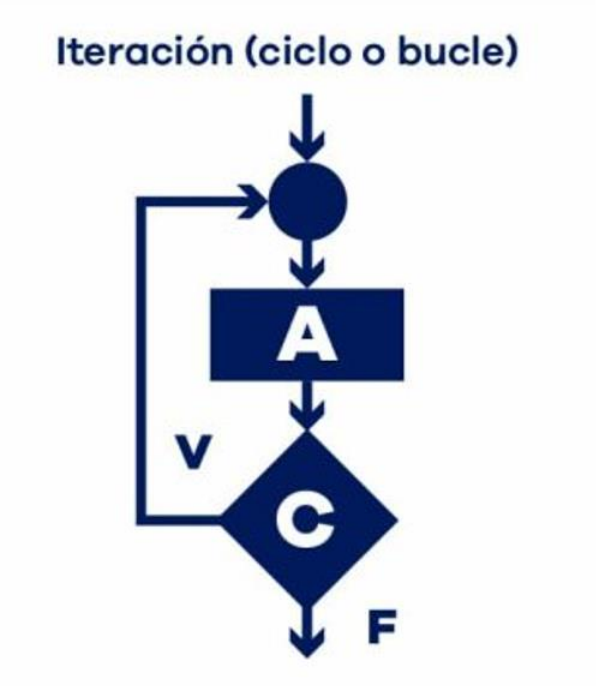
O número de casos que podemos acrescentar a este tipo de condicional é infinito.

E podemos até mesmo fazer condicionantes aninhados uns dentro dos outros, de modo que a complexidade aumenta 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**

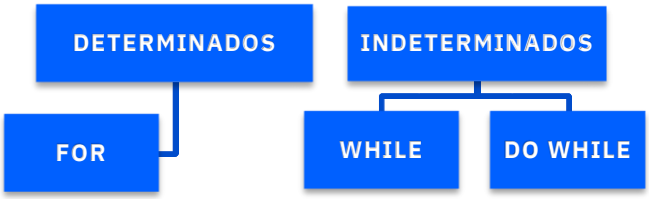
```
Inicio
    LEER a
    LEER b
    LEER S
    Si S='+'
        IMPRIMIR 'El resultado es =', a+b
    Si no, si S='-'
        IMPRIMIR 'El resultado es =', a-b
    Si no, si S='*'
        IMPRIMIR 'El resultado es =', a*b
    Si no, si b=0
        Si a=0
            IMPRIMIR 'El resultado es =', NaN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

Laços

Elas permitem que um conjunto de instruções seja executado repetidamente, seja um número pré-determinado de vezes, ou até que uma determinada condição seja verificada.



Há dois tipos de loops, dependendo se sabemos ou não o número de repetições que nosso loop irá fazer.



Nos loops determinados, o número de vezes que ele será repetido é conhecido com certeza, por exemplo, podemos colocar uma frase em loop e ela será mostrada na tela um determinado número de vezes.

Em loops indeterminados, não sabemos ao certo quantas vezes um loop se repetirá, já que será repetido até que se cumpra uma condição que, a priori, não sabemos quando será cumprida.

Por exemplo, imprimir repetidamente uma frase na tela até que o usuário pressione uma tecla para parar o loop.

As diferentes estruturas de controle são descritas abaixo. Para cada um deles, é descrito o fluxograma e uma sintaxe genérica, que será ligeiramente diferente dependendo do idioma que estamos usando, mas basicamente em todos os idiomas de alto nível é muito semelhante.

Note que todos eles têm uma única entrada e uma única saída.

Estrutura de repetição indexada: FOR

Este tipo de estrutura permite implementar a repetição de um determinado conjunto de instruções um número pré-determinado de vezes.

Isto é feito usando uma variável de controle de laço, também chamada de índice, que percorre um conjunto pré-fixado de valores em uma determinada ordem. Para cada valor de índice desse conjunto, o mesmo conjunto de instruções é executado uma vez.

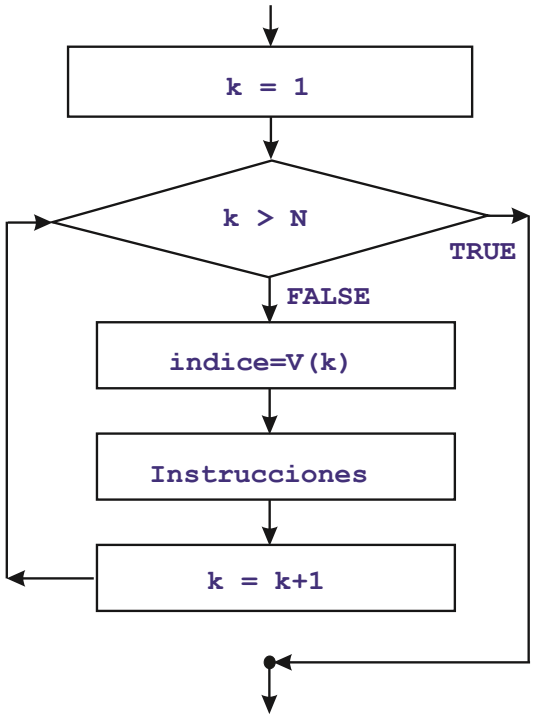
No exemplo a seguir, o bloco de instruções é executado uma vez para cada valor do índice, que sucessivamente toma o valor de cada componente do vetor V, de comprimento N.

... **for** indice=V

instruções

...

El



índice dos laços de laço através dos valores de um vetor V de comprimento N.

Como exemplo de utilização da estrutura FOR, veja o seguinte pseudo-código para calcular a soma dos primeiros números ímpares.

- O valor da variável de controle do índice pode ou não ser usado dentro do conjunto de instruções que faz parte do corpo FOR, mas não deve ser modificado.
- O conjunto de valores a serem percorridos pelo índice pode estar **vazio** (N=0). Neste caso, o bloco de instruções não é executado em absoluto.
- As estruturas FOR e IF podem ser "aninhadas", ou seja, incluir uma dentro da outra, com a restrição (senso comum) de que a interna tem que estar completamente contida em um dos blocos de instruções da outra.

Dado um número inteiro, n, calcular a soma dos primeiros números n ímpares.

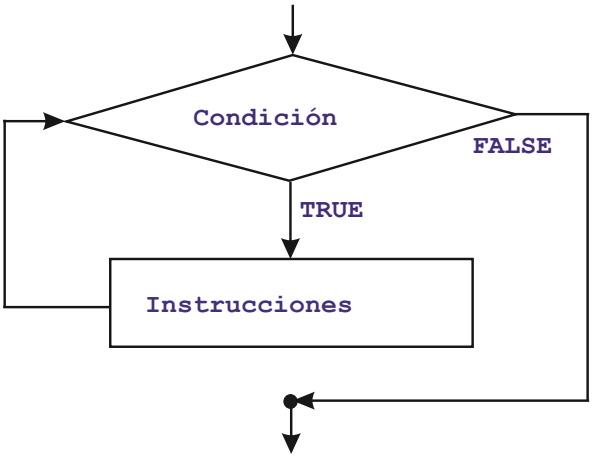
```
Inicio
    LEER n
    HACER suma=0
    Para i= 1, 3, 5, ..., 2*n-1 HACER suma=suma+i
    Fin Para
    IMPRIMIR 'La suma vale : ', suma
Fin
```

Algoritmo 5.10 Dado um número natural, n, imprima a lista de seus divisores, em ordem decrescente.

```
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
```

Estrutura repetitiva condicional: WHILE

Ela permite implementar a repetição do mesmo conjunto de instruções desde que uma determinada condição seja verificada: o número de vezes que o ciclo será repetido não é definido a priori. O fluxograma descritivo desta estrutura é mostrado na figura a seguir.



...

while expression-logic

instruções

end

...

Seu funcionamento é evidente a partir do diagrama:

- No início de cada iteração, a expressão lógica é avaliada.
- Se o resultado for VERDADEIRO, o conjunto de instruções é executado e iteratiza novamente, ou seja, o passo 1 é repetido.
- Se o resultado for FALSO, a execução do loop WHILE é interrompida e o programa continua a ser executado para a instrução que segue o END.

A seguir, um exemplo do uso desta estrutura.

Imprimir os primeiros 100 números naturais em ordem ascendente.

Inicio i=1

Mientras que i ≤ 100

IMPRIMIR i

HACER i=i+1

Fin Mientras Fin

Quebra de ciclos de repetição: BREAK y CONTINUE

Às vezes é necessário interromper a execução de um ciclo de repetição **em algum ponto dentro do bloco de instruções que está sendo repetido**.

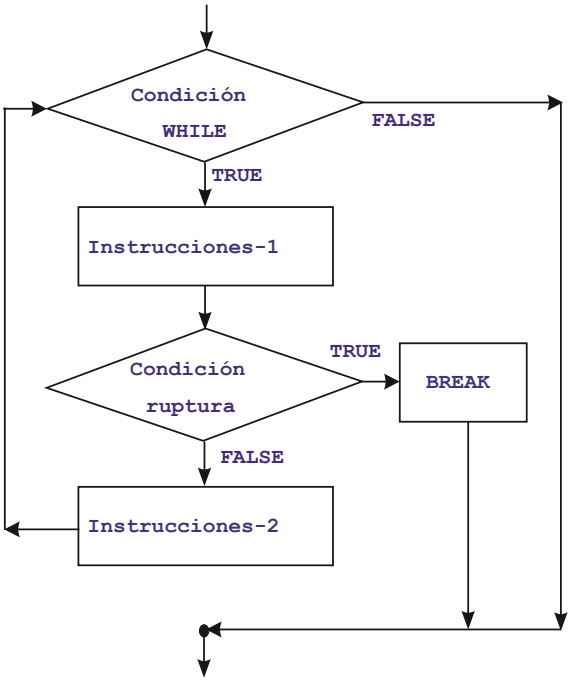
Logicamente, isto dependerá se alguma condição for verificada ou não.

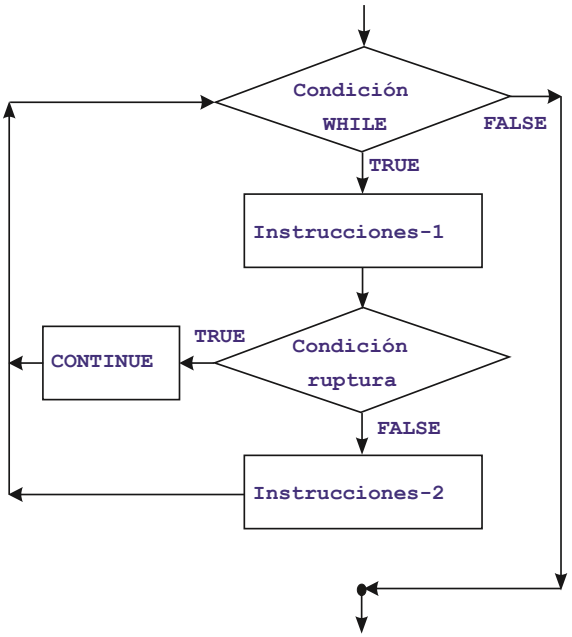
A interrupção pode ser feita de duas maneiras:

1. Abandonar o ciclo de repetição para sempre.
2. Abandonando a atual iteração, mas iniciando a próxima.

As instruções para implementar isto têm diferentes nomes em diferentes linguagens de programação. Em Javascript, a primeira opção é implementada com a instrução BREAK e a segunda com a instrução CONTINUE. Ambos podem ser usados para quebrar um loop FOR ou WHILE. Quando o comando BREAK é usado dentro de um loop FOR, o índice do loop retém, fora do loop, o último valor que ele tomou.

Veja os fluxogramas correspondentes nas figuras a seguir.





Estrutura de escolha entre vários casos: SWITCH

Este tipo de estrutura torna possível decidir entre vários caminhos possíveis, dependendo do valor tomado por uma determinada instrução.

Nem todas as linguagens de programação possuem esta estrutura.

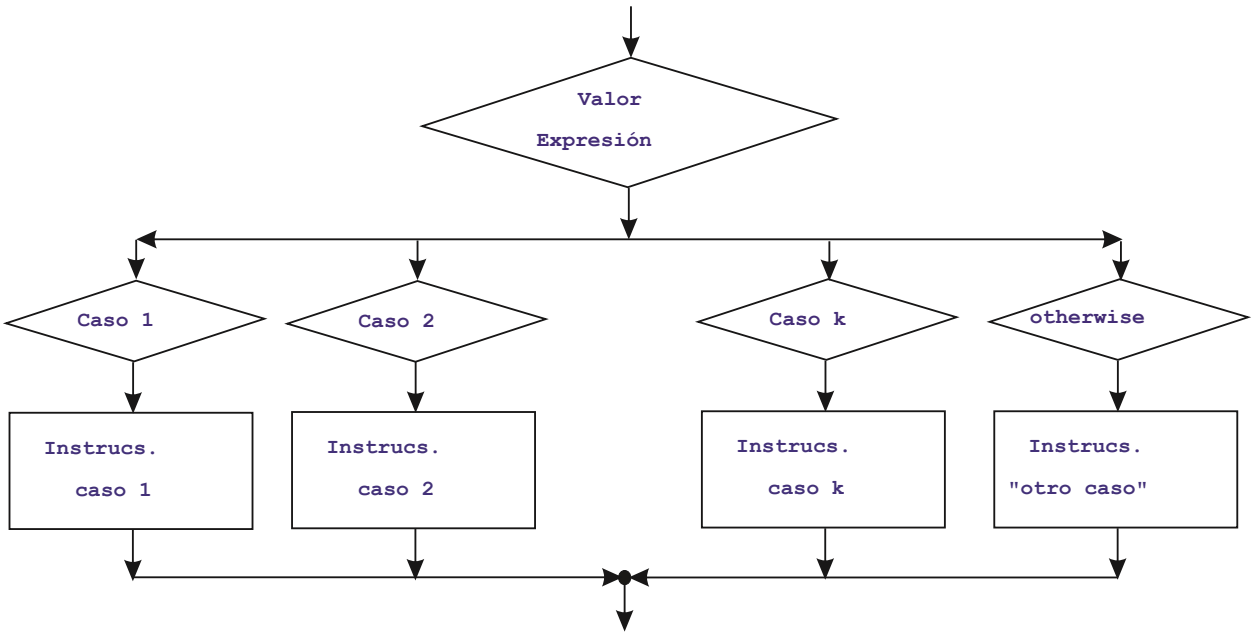
O diagrama de fluxo correspondente a uma destas estruturas (com quatro casos) é apresentado na Figura 5.12.

switch expresión

case valor-1 instrucciones caso 1

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

default instruções de casos diferentes
end



Em cada caso, o valor correspondente pode ser ou um único valor ou um conjunto de valores, caso em que estão entre parênteses. A cláusula DEFAULT e seu conjunto de instruções correspondentes podem não estar presentes.

A operação é a seguinte:

- No início, a expressão é avaliada.
- Se a expressão toma o valor (ou valores) especificado ao lado da primeira cláusula CASE, o conjunto de instruções deste caso é executado e então a estrutura SWITCH é abandonada, continuando com a instrução seguindo a END.
- O procedimento acima é repetido, em ordem, para cada uma das cláusulas do CASE que se seguem.
- Se a cláusula DEFAULT estiver presente e a expressão não tiver tomado nenhum dos valores especificados acima, o conjunto de instruções correspondente é executado.

Observe que no máximo o conjunto de instruções de um dos casos é executado, ou seja, uma vez verificado um caso e executado seu conjunto de instruções, o resto dos casos não são testados, uma vez que a estrutura é abandonada. Obviamente, se a cláusula DEFAULT não estiver presente, nenhum dos casos poderá ocorrer.

Exemplo do uso da declaração SWITCH.

Dados dois números reais, **a e b**, e o símbolo, **S** (personagem), de um operador aritmético (**+**, **-**, *****, **/**), imprimir o resultado da operação **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 as estruturas de controle (também chamadas de modificadores de fluxo de execução) apresentadas neste tópico são genéricas. A maioria das linguagens de programação as tem, mas não todas, e sua sintaxe difere ligeiramente de uma para outra.

Mais tarde, neste curso, veremos aqueles que correspondem ao *Javascript* e *Python*.