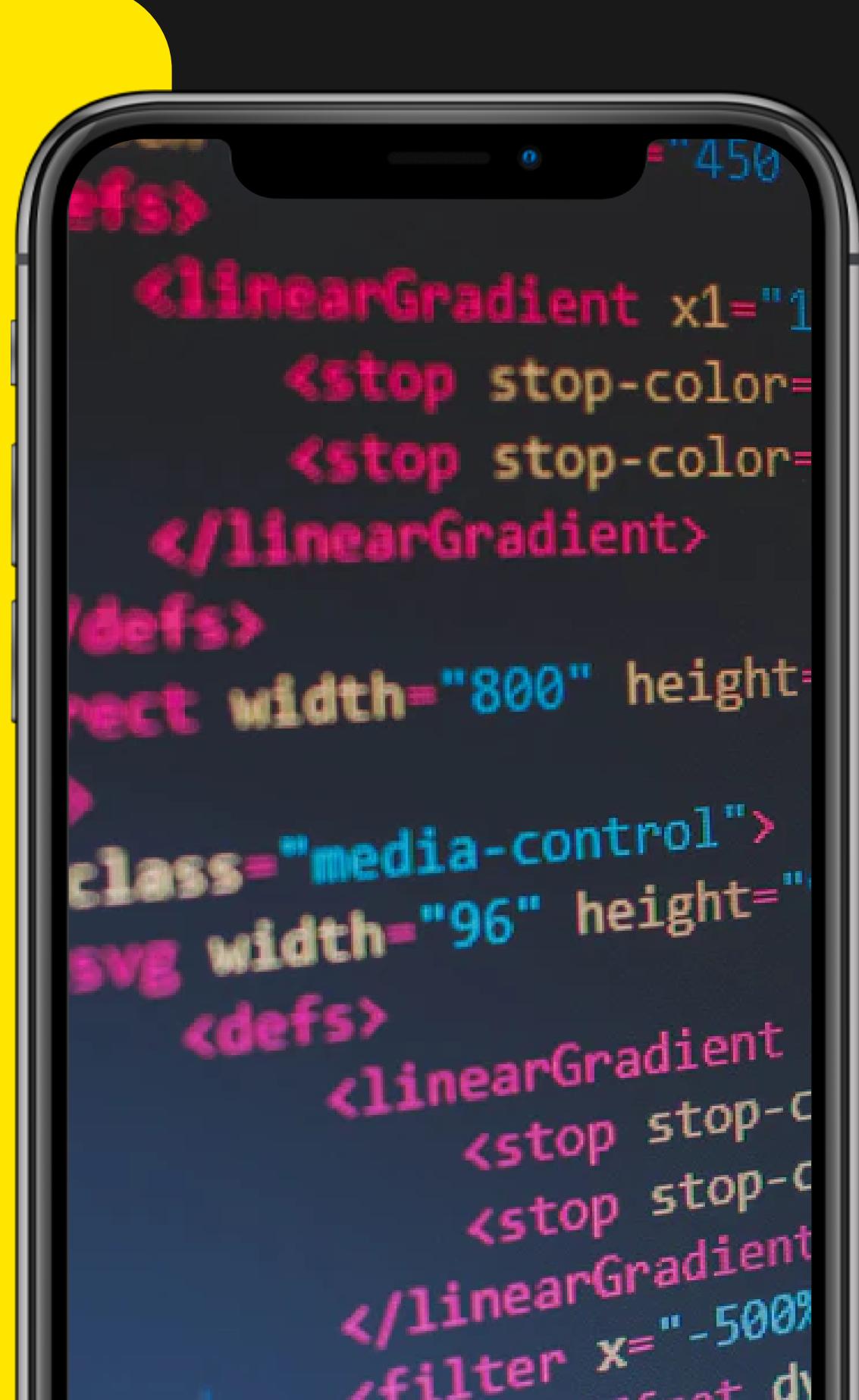


PROGRAMACIÓN DE COMPUTADORAS 2

Clase 03

→ Sentencias de Control



Contenido



Tipo de Sentencia de Control



If / If-Else / If-ElseIf



Select-Case



Funciones Básicas



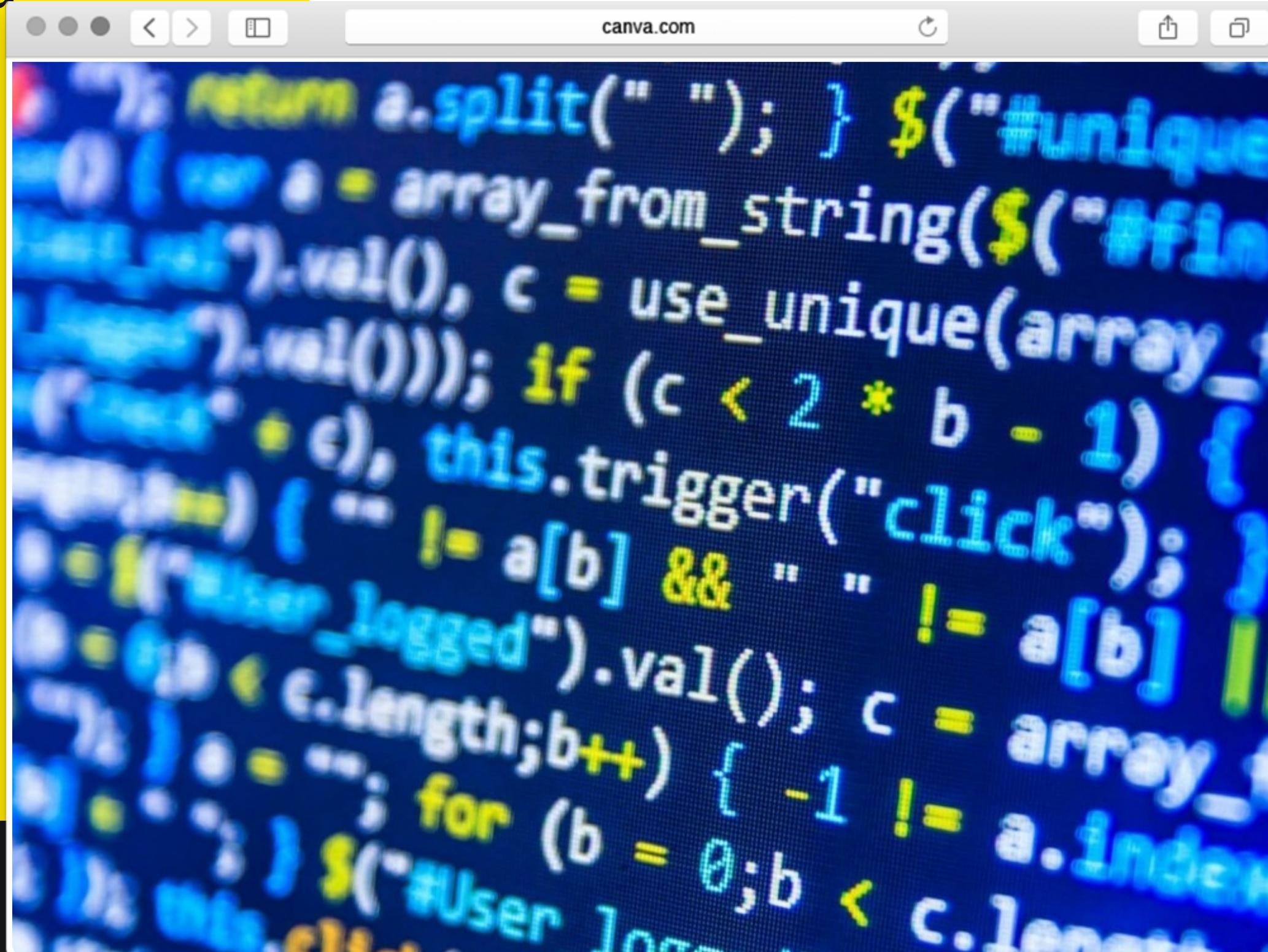
Ejemplo

Sentencias de control del flujo

Hasta ahora un programa no era más que una secuencia de instrucciones que son ejecutadas en el orden en que son formuladas. Además, si quisieramos que un conjunto de ellas se repitiera varias veces, no tenemos más remedio que escribirlas varias veces.



El orden de ejecución de las sentencias es lo que se conoce por el flujo del programa, y se puede variar a voluntad utilizando las sentencias de control de flujo, con las que es muy fácil alterarlo para adecuarlo a nuestras necesidades. Esto es lo que hace potentes a los lenguajes de programación. Básicamente, existen tres tipos de sentencias, secuencial, condicional e iterativa, y con ellas se puede escribir cualquier programa.



The image shows a screenshot of a web browser window with a blurred background of code. The browser's address bar at the top right displays 'canva.com'. The main content area is filled with a dense, multi-colored pattern of letters and symbols, appearing as if it were a large block of programming code. The colors include shades of blue, green, yellow, and white, typical of syntax-highlighted code. The overall effect is abstract due to the blurring, but the structure of a code editor is visible.

Secuencia

Es una serie de pasos.
Puede ser solo un paso

Paso 1

Paso 2

- .
- .
- .
- .

Paso n

Selección simple → 11

Se especifica si una condición es verdadera se hace la primer secuencia de paso, si no se realiza la otra

si condición entonces
secuencia1

si condición entonces
secuencia1

sino
secuencia2

Repetir

Permite llevar a cabo la secuencia de pasos al menos una vez, repitiéndola hasta que se cumpla la condición

repetir

secuencia

mientras condición sea verdadera
ó

repetir

secuencia

hasta condición sea verdadera

Selección simple

Repite una secuencia de pasos, mientras se cumpla la condición

mientras condición sea verdadera **hacer**
secuencia

Para

Repite una cantidad fija de veces, de acuerdo a los elementos del conjunto

para cada elemento de un conjunto ***hacer***
secuencia

Tambien se le puede indicar la cantidad de veces que debe repetirse

para contador = 1 ***hasta*** 10 ***con incremento*** 1
hacer

Sentencias Condicionales



Existen dos formas de alterar el flujo dependiendo de una condición. La idea básica es la de encontrarnos ante una bifurcación de un camino, en la que seguiremos por uno u otro camino dependiendo de la respuesta a una pregunta que se halla escrita en la bifurcación. Una variante de esto es que en lugar de dos posibilidades, se nos presenten más caminos por los que poder seguir.





If Simple

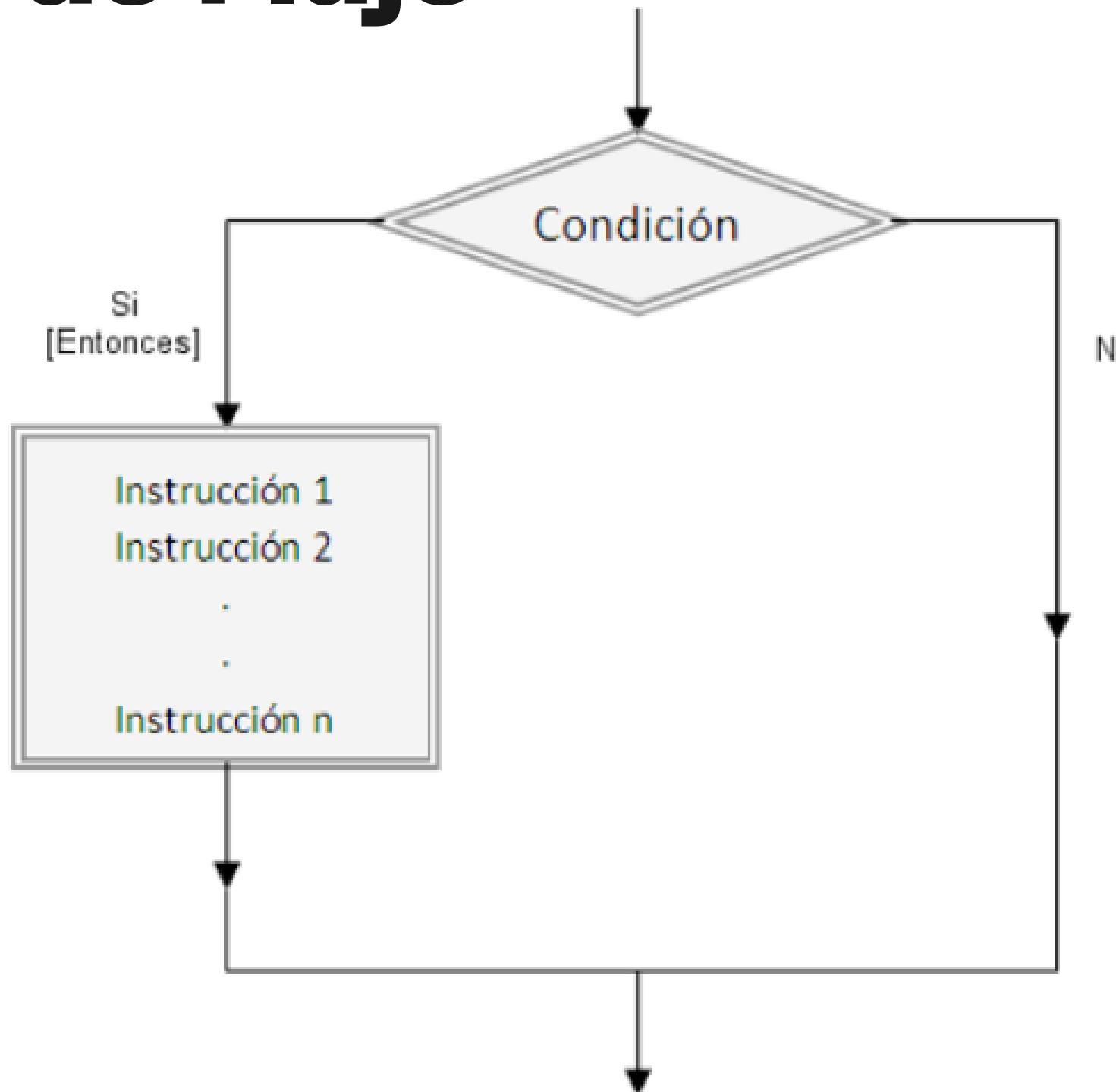
Es la forma más elemental de control de flujo condicional. Una sentencia if evaluará primero su condición, y si se evalúa como true ejecutará la sentencia o bloque de sentencias que se encuentre justo después de la condición.

```
if(Condición){  
    acción;  
}
```



If - Else

Diagrama de Flujo





If - Else

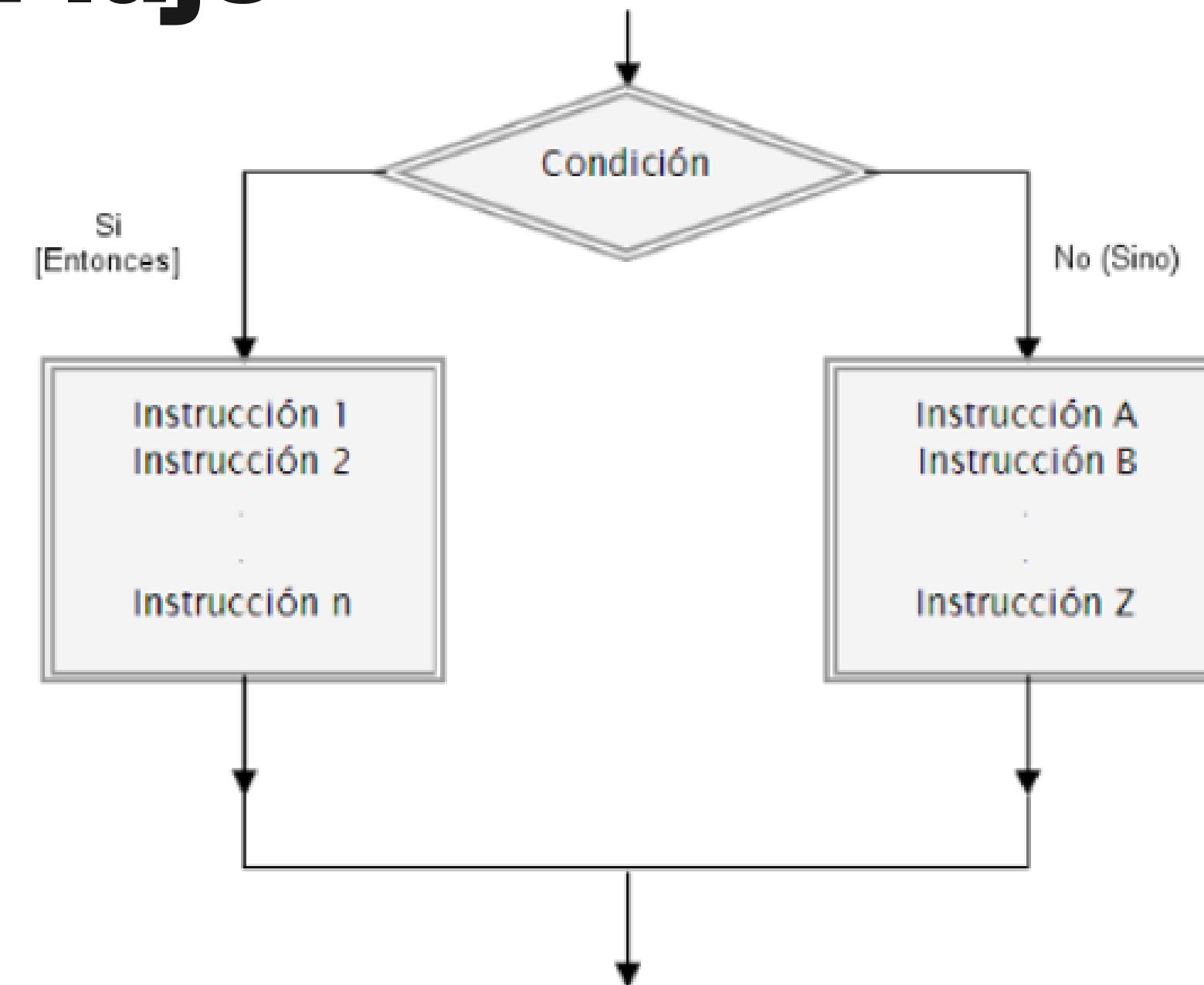
Si la condición se evalúa como false y existe una cláusula else, entonces se ejecutará la sentencia o bloque de sentencias que siguen al else. Esta parte es opcional. La única restricción que se impone a la expresión condicional es que sea una expresión válida de tipo booleano que se evalúe como true o como false.

```
if (expresión_condicional)
    sentencia1
else
    sentencia2
```



If - Else

Diagrama de Flujo





If - Else

Sintaxis

```
' Multiline syntax:  
If condition [ Then ]  
    [ statements ]  
    [ ElseIf elseifcondition [ Then ]  
        [ elseifstatements ] ]  
    [ Else  
        [ elsestatements ] ]  
End If  
  
' Single-line syntax:  
If condition Then [ statements ] [ Else [ elsestatements ] ]
```



If - Else

Ejemplo

Se elige el mayor de dos enteros

```
Dim a As Integer = 3
Dim b As Interger = 5
Dim m As Integer

If a < b Then
    m = b
else
    m = a
End IF
```



If - Else

Ejemplo

Un ejemplo de condicional sin parte else es el siguiente, donde se imprime si un número es par o impar

```
Dim a As Integer = 3
Dim mensaje As String = "El número " + a + " es "

If a % 2 != 0 Then
    mensaje = mensaje + "impar"
End IF

mensaje = mensaje + "par."
' El numero 3 es impar
```



If - Else-If

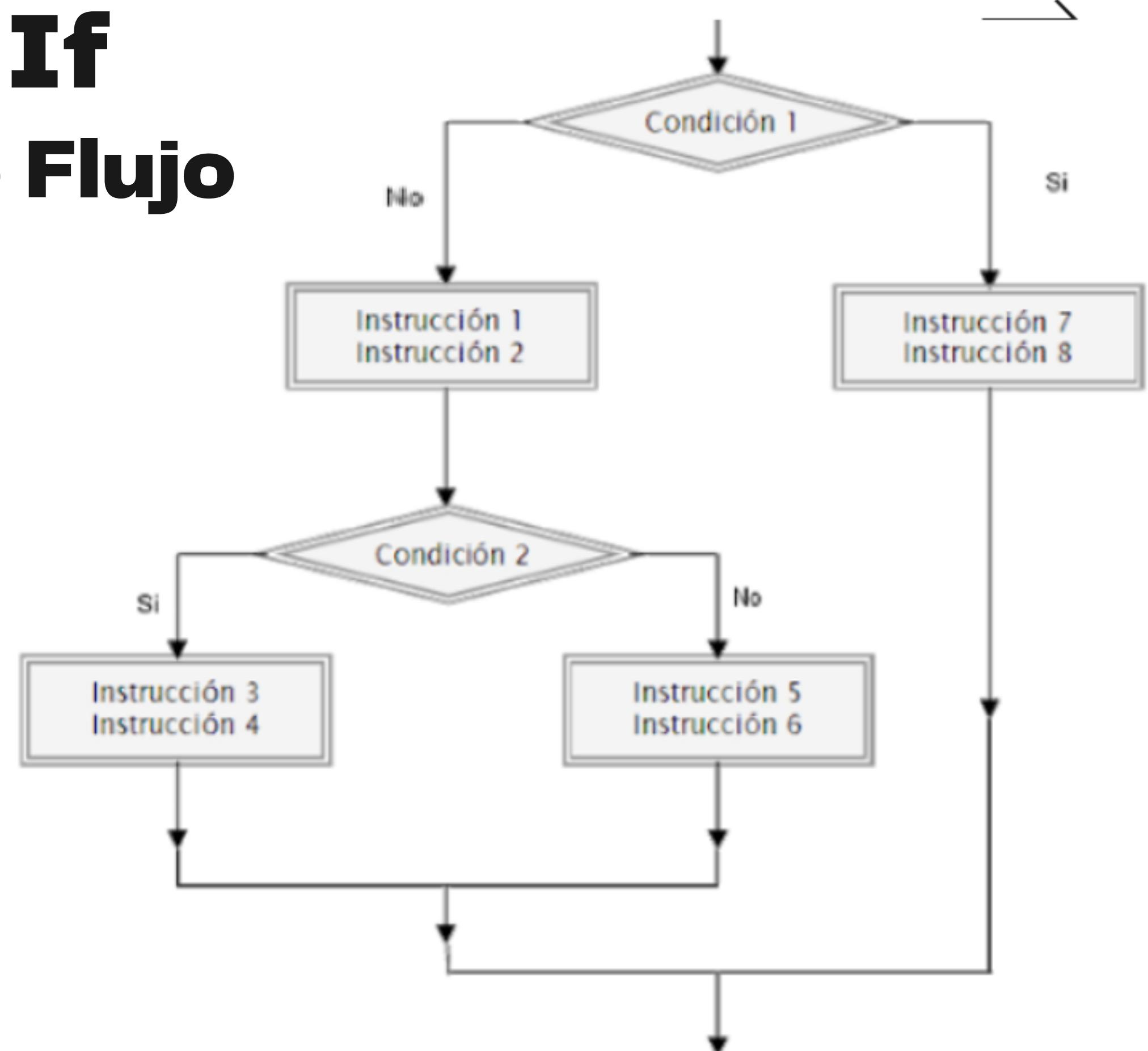
Ejemplo

Es posible construir una serie de comprobaciones uniendo un if a la cláusula else de un if anterior. Por ejemplo, el siguiente trozo de código muestra una calificación dada una nota numérica

```
Dim nota As float = 6.7F  
Dim calificacion As String  
  
If nota >= 9.0 Then  
    calificacion="Sobresaliente"  
ElseIf nota >= 7.0 Then  
    calificacion="Notable"  
ElseIf nota >= 5.0 Then  
    calificacion="Aprobado"  
Else  
    calificacion="Suspendido"  
End IF
```

If - Else - If

Diagrama de Flujo



Module Multiline

```
Public Sub Main()
    'Create a Random object to seed our starting value
    Dim randomizer As New Random()
    'set our variable
    Dim count As Integer = randomizer.Next(0, 5)

    Dim message As String

    'If count is zero, output will be no items
    If count = 0 Then
        message = "There are no items."
    'If count is 1, output will be "There is 1 item.".
    ElseIf count = 1 Then
        message = "There is 1 item."
    'If count is greater than 1, output will be "There are {count} items.",
    'where {count} is replaced by the value of count.
    Else
        message = $"There are {count} items."
    End If

    Console.WriteLine(message)
End Sub
End Module
'This example displays output like the following:
' There are 4 items.
```





Select - Case

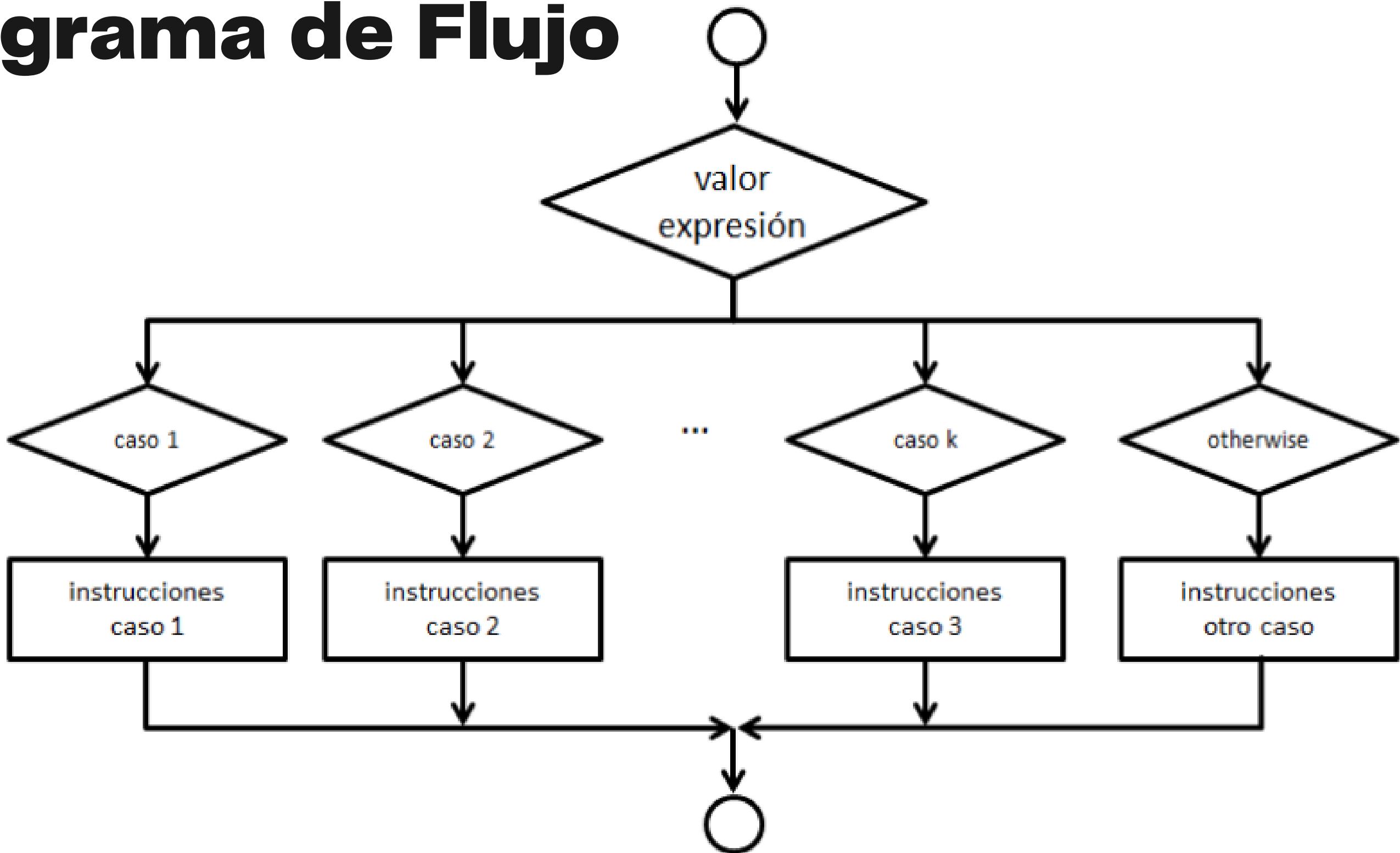
Mediante esta instrucción establecemos una serie de casos que se podrían cumplir para el valor de una expresión o variable, y en función del valor de la variable se ejecutarán una serie de instrucciones asociadas u otra.

```
Select Case [expresión]
Case [valor expresión 1]
    Instrucción 1
    Instrucción 2
Case [valor expresión 2]
    Instrucción 3
    Instrucción 4
    .
    .
    .
Case [valor expresión n]
    Instrucción k
Case Else
    Instrucción m
End Select
```



Select - Case

Diagrama de Flujo





Select - Case

La expresión a evaluar puede ser un valor numérico o una cadena de texto. Sólo se puede evaluar una expresión y no múltiples expresiones. La evaluación de expresiones puede ser:

- **De coincidencia:** por ejemplo, Case 12 indicaría que si la expresión evaluada vale 12 se ejecutarán las instrucciones anexas.
- **De intervalo:** usando la palabra clave To. Por ejemplo, Case 12 To 14 indicaría que si la expresión evaluada tiene un valor comprendido entre 12 y 14 (incluidos los extremos de los intervalos), se ejecutarán las instrucciones anexas.
- **De comparación:** usando la palabra clave Is. Por ejemplo, Case Is <= 14 indicaría que si la expresión evaluada tiene un valor menor o igual a 14 se ejecutarán las instrucciones anexas.

Funciones básicas

Visual Basic cuenta con diferentes funciones por defecto que facilitan el trabajo a la hora de escribir código. Existen diferentes tipos de funciones

- Funciones Matemáticas
- Funciones de Conversión
- Funciones de Conversión de Tipo
- Funciones varias

Documentación detallada:

<https://docs.microsoft.com/es-es/office/vba/language/reference/functions-visual-basic-for-applications>

Funciones Matemáticas



Abs, Pow, Sqr	Trigonométricas	Log y Exp	Rnd	Sgn
Devuelve el valor absoluto, una potencia, y una raíz cuadrada de un valor numérico respectivamente.	Incluye todas las que derivan de la tangente, seno, y coseno.	Si es Log, devuelve el logaritmo natural de la base que se especifique. Si es EXP devuelve el "e" elevado al número especificado.	Devuelve un valor pseudo aleatorio. Se requiere de una fórmula para establecer mínimo y máximo. $\text{Int}((\text{max} * \text{Rnd}) + \text{min})$	Devuelve el signo que posee un valor numérico.

Funciones Matemáticas



Abs, Pow, Sqr

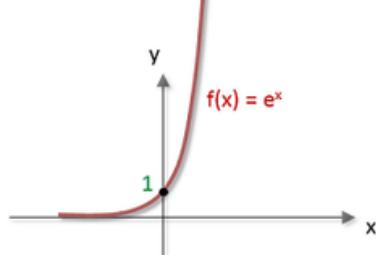
```
Dim MyNumber  
MyNumber = Abs(50.3)      ' Returns 50.3.  
MyNumber = Abs(-50.3)     ' Returns 50.3.
```

```
Math.Pow(Number,Power)
```

```
Dim MySqr  
MySqr = Sqr(4)           ' Returns 2.  
MySqr = Sqr(23)          ' Returns 4.79583152331272.  
MySqr = Sqr(0)           ' Returns 0.  
MySqr = Sqr(-4)          ' Generates a run-time error.
```

Exp

```
Dim MyAngle, MyHSin  
' Define angle in radians.  
MyAngle = 1.3  
' Calculate hyperbolic sine.  
MyHSin = (Exp(MyAngle) - Exp(-1 * MyAngle)) / 2
```



Trigonométricas

Secant Sec(X) = 1

Cosecant Cosec(X) =

Cotangent Cotan(X) =

Seno inverso Arcsin(X)

Lista completa de funciones

<https://docs.microsoft.com/es-es/office/vba/language/reference/user-interface-help/derived-math-functions>

Log

```
Static Function Log10(X)  
    Log10 = Log(X) / Log(10#)  
End Function
```

```
Dim MyAngle, MyLog  
' Define angle in radians.  
MyAngle = 1.3  
' Calculate inverse hyperbolic sine.  
MyLog = Log(MyAngle + Sqr(MyAngle * MyAngle + 1))
```

Rnd

```
Dim MyValue As Integer  
MyValue = Int((6 * Rnd) + 1)
```

' Generate random value between 1 and 6.

Sgn

```
Dim MyVar1, MyVar2, MyVar3, MySign  
MyVar1 = 12: MyVar2 = -2.4: MyVar3 = 0  
MySign = Sgn(MyVar1)           ' Returns 1.  
MySign = Sgn(MyVar2)           ' Returns -1.  
MySign = Sgn(MyVar3)           ' Returns 0.
```



Funciones Conversión

Asc / Chr	CVErr	Format	Str	Val
<ul style="list-style-type: none">Asc: devuelve el valor ascii de un carácter.Chr: devuelve el carácter de un valor ascii.	Sirve crear errores definidos por el usuario en procedimientos creados por el usuario.	Define el formato que se le indique a una variable, principalmente a variables de tipo fecha.	Devuelve una representación de Variant (String) de un número.	Devuelve los números incluidos en una cadena como un valor numérico del tipo apropiado.

Funciones Matemáticas



Asc, Chr

```
Dim MyNumber  
MyNumber = Asc("A")      ' Returns 65.  
MyNumber = Asc("a")      ' Returns 97.  
MyNumber = Asc("Apple")   ' Returns 65.
```

```
Dim MyChar  
MyChar = Chr(65)        ' Returns A.  
MyChar = Chr(97)        ' Returns a.  
MyChar = Chr(62)        ' Returns >.  
MyChar = Chr(37)        ' Returns %.
```

CVErr

```
' Call CalculateDouble with an error-producing argument.  
Sub Test()  
    Debug.Print CalculateDouble("345.45robert")  
End Sub  
' Define CalculateDouble Function procedure.  
Function CalculateDouble(Number)  
    If IsNumeric(Number) Then  
        CalculateDouble = Number * 2      ' Return result.  
    Else  
        CalculateDouble = CVErr(2001)      ' Return a user-defined error  
    End If    ' number.  
End Function
```

Format

```
Dim MyTime, MyDate, MyStr  
MyTime = #17:04:23#  
MyDate = #January 27, 1993#  
  
' Returns current system time in the system-defined long time format.  
MyStr = Format(Time, "Long Time")  
  
' Returns current system date in the system-defined long date format.  
MyStr = Format(Date, "Long Date")  
  
MyStr = Format(MyTime, "h:m:s")    ' Returns "17:4:23".  
MyStr = Format(MyTime, "hh:mm:ss am/pm")  ' Returns "05:04:23 pm".  
MyStr = Format(MyTime, "hh:mm:ss AM/PM")   ' Returns "05:04:23 PM".  
MyStr = Format(MyDate, "dddd, mmm d yyyy")  ' Returns "Wednesday, Jan 27 1993".  
' If format is not supplied, a string is returned.  
MyStr = Format(23)      ' Returns "23".  
  
' User-defined formats.  
MyStr = Format(5459.4, "#,##0.00")    ' Returns "5,459.40".  
MyStr = Format(334.9, "###0.00")      ' Returns "334.90".  
MyStr = Format(5, "0.00%")           ' Returns "500.00%".  
MyStr = Format("HELLO", "<")        ' Returns "hello".  
MyStr = Format("This is it", ">")    ' Returns "THIS IS IT".
```

Str

```
Dim MyString  
MyString = Str(459)      ' Returns " 459".  
MyString = Str(-459.65)   ' Returns "-459.65".  
MyString = Str(459.001)   ' Returns " 459.001".
```

Val

```
Dim MyValue  
MyValue = Val("2457")    ' Returns 2457.  
MyValue = Val(" 2 45 7")  ' Returns 2457.  
MyValue = Val("24 and 57") ' Returns 24.
```

Funciones de Conversión de Tipo



- CBool(expression)
- CByte(expression)
- CCur(expression)
- CDate(expression)
- CDbl(expression)
- CDec(expression)
- Clnt(expression)
- CLng(expression)
- CLngLng(expression) (válida solo en plataformas de 64 bits).
- CLngPtr(expression)
- CSng(expression)
- CStr(expression)
- CVar(expression)

```
Dim MyDate, MyShortDate, MyTime, MyShortTime  
MyDate = "February 12, 1969" ' Define date.  
MyShortDate = CDate(MyDate) ' Convert to Date data type.  
  
MyTime = "4:35:47 PM" ' Define time.  
MyShortTime = CDate(MyTime) ' Convert to Date data type.
```

```
Dim A, B, Check  
A = 5: B = 5 ' Initialize variables.  
Check = CBool(A = B) ' Check contains True.  
  
A = 0 ' Define variable.  
Check = CBool(A) ' Check contains False.
```

```
Dim MyDouble, MyByte  
MyDouble = 125.5678 ' MyDouble is a Double.  
MyByte = CByte(MyDouble) ' MyByte contains 126.
```

```
Dim MyDouble, MyCurr  
MyDouble = 543.214588 ' MyDouble is a Double.  
MyCurr = CCur(MyDouble * 2) ' Convert result of MyDouble * 2  
' (1086.429176) to a  
' Currency (1086.4292).
```

```
Dim MyDouble, MyInt  
MyDouble = 2345.5678 ' MyDouble is a Double.  
MyInt = CInt(MyDouble) ' MyInt contains 2346.
```

Funciones Varias



- IsArray
- IsDate
- IsEmpty
- IsError
- IsMissing
- IsNull
- IsNumeric
- IsObject
- LTrim, RTrim y Trim

```
Dim MyString, TrimString  
MyString = " <-Trim-> " ' Initialize string.  
TrimString = LTrim(MyString) ' TrimString = "<-Trim-> ".  
TrimString = RTrim(TrimString) ' TrimString = " <-Trim->".  
TrimString = LTrim(RTrim(TrimString)) ' TrimString = "<-Trim->".  
' Using the Trim function alone achieves the same result.  
TrimString = Trim(MyString) ' TrimString = "<-Trim->".
```

```
Dim ReturnValue  
' The following statements call the user-defined function  
ReturnValue = ReturnTwice() ' Returns Null.  
ReturnValue = ReturnTwice(2) ' Returns 4.  
  
' Function procedure definition.  
Function ReturnTwice(Optional A)  
    If IsMissing(A) Then  
        ' If argument is missing, return a Null.  
        ReturnTwice = Null  
    Else  
        ' If argument is present, return twice the value.  
        ReturnTwice = A * 2  
    End If  
End Function
```

```
Dim MyArray(1 To 5) As Integer, YourArray, MyCheck  
YourArray = Array(1, 2, 3) ' Use Array function.  
MyCheck = IsArray(MyArray) ' Returns True.  
MyCheck = IsArray(YourArray) ' Returns True.
```

```
Dim MyVar, MyCheck  
MyVar = "53" ' Assign value.  
MyCheck = IsNumeric(MyVar) ' Returns True.  
  
MyVar = "459.95" ' Assign value.  
MyCheck = IsNumeric(MyVar) ' Returns True.  
  
MyVar = "45 Help" ' Assign value.  
MyCheck = IsNumeric(MyVar) ' Returns False.
```

```
Dim MyVar, MyCheck  
MyVar = "04/28/2014" ' Assign valid date value.  
MyCheck = IsDate(MyVar) ' Returns True.  
  
MyVar = "April 28, 2014" ' Assign valid date value.  
MyCheck = IsDate(MyVar) ' Returns True.  
  
MyVar = "13/32/2014" ' Assign invalid date value.  
MyCheck = IsDate(MyVar) ' Returns False.  
  
MyVar = "04.28.14" ' Assign valid time value.  
MyCheck = IsDate(MyVar) ' Returns True.  
  
MyVar = "04.28.2014" ' Assign invalid time value.  
MyCheck = IsDate(MyVar) ' Returns False.
```

```
Dim ReturnVal, MyCheck  
ReturnVal = UserFunction()  
MyCheck = IsError(ReturnVal) ' Returns True.
```

```
Dim MyVar, MyCheck  
MyCheck = IsEmpty(MyVar) ' Returns True.  
  
MyVar = Null ' Assign Null.  
MyCheck = IsEmpty(MyVar) ' Returns False.  
  
MyVar = Empty ' Assign Empty.  
MyCheck = IsEmpty(MyVar) ' Returns True.
```