

# Introducción a Quality Assurance

## Bases de programación

### ¿Qué es la Programación?

En informática el término programación se refiere a la acción de crear programas y programar es la serie de instrucciones, que le vamos a dar a nuestra computadora para lograr que nuestro programa funcione.

Las partes que componen a nuestro programa son el lenguaje de programación y los algoritmos.

### ¿Qué es un lenguaje de programación?

Es un lenguaje formal que, mediante una serie de instrucciones, le permite a un programador escribir un conjunto de órdenes, acciones consecutivas, datos y algoritmos para, de esa forma, resolver problemas.

Las instrucciones que sigue la computadora para la creación de programas están escritas en un lenguaje de programación y luego son traducidas a un lenguaje de máquina que puede ser interpretado y ejecutado por el hardware del equipo.

### ¿Qué es un algoritmo?

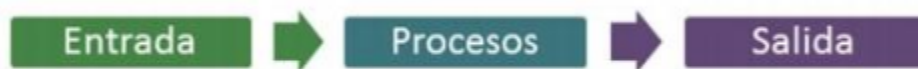
Las instrucciones que le vamos a dar a nuestro programa, se conocen como algoritmos. Un algoritmo es un método para darle instrucciones a nuestro programa y resolver un problema.

Este consiste en la realización de un conjunto de pasos lógicamente ordenados tal que, partiendo de la información que le demos, permite obtener ciertos resultados que conforman la solución del problema.

Los algoritmos son independientes tanto del lenguaje de programación en que se expresan como de la computadora que los ejecuta. En cada problema el algoritmo se puede expresar en un lenguaje diferente de programación y ejecutarse en una computadora distinta; sin embargo, el algoritmo será siempre el mismo. Así, por ejemplo, en una analogía con la vida diaria, una receta de un plato de cocina se puede expresar en español, inglés o francés, pero cualquiera que sea el lenguaje, los pasos para la elaboración del plato se realizan sin importar el idioma del cocinero.

Los algoritmos son más importantes que los lenguajes de programación o las computadoras. Un lenguaje de programación es sólo un medio para expresar un algoritmo y una computadora es solo un procesador para ejecutarlo. Tanto el lenguaje de programación como la computadora son los medios para obtener un fin: conseguir que el algoritmo se ejecute y se efectúe el proceso correspondiente.

El programador debe constantemente resolver problemas de manera algorítmica, lo que significa plantear el problema de forma tal que queden indicados los pasos necesarios para obtener los resultados pedidos, a partir de los datos conocidos. Lo anterior implica que un algoritmo básicamente consta de tres elementos: Datos de Entrada o Información de entrada, Procesos y la Información de Salida.



Estructura de un Programa: Datos de entrada, proceso y salida.

## Programa

¿Dónde se van a ver reflejados los lenguajes de programación y los algoritmos? En nuestro programa.

Un programa no es más que una serie de algoritmos escritos en algún lenguaje de programación de computadoras. Un programa es, por lo tanto, un conjunto de instrucciones

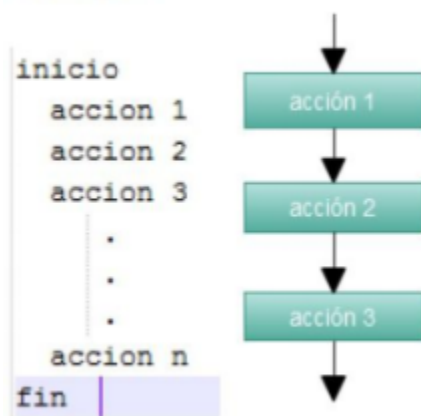
—órdenes dadas a la computadora— que producirán la ejecución de una determinada tarea. En esencia, un programa es un medio para conseguir un fin. El fin será probablemente definido como la información necesaria para solucionar un problema.

## Especificaciones de un programa

Tras la decisión de desarrollar un programa, el programador debe establecer el conjunto de especificaciones que debe contener el programa: entrada, salida y algoritmos de resolución, que incluirán las técnicas para obtener las salidas a partir de las entradas.

Un programa puede ser lineal (secuencial) o no lineal. Un programa es lineal si las instrucciones (acciones) se ejecutan secuencialmente como los ejercicios propuestos en esta guía, es decir, sin bifurcaciones, decisión ni comparaciones.

### Pseudocódigo



Estructura de un programa secuencial

## Codificación

Una vez que tenemos las especificaciones de un programa pasaremos a la codificación del programa. La codificación es la operación de escribir la solución del problema (de acuerdo a la lógica del pseudocódigo), en una serie de instrucciones detalladas, en un código reconocible por la computadora. La serie de instrucciones detalladas se conoce como código fuente, el cual se escribe en un lenguaje de programación o lenguaje de alto nivel.

## ¿Cómo deben escribirse los algoritmos/programas?

Ya sabemos que es un programa, el diseño de un programa, las especificaciones de un programa y su codificación. Ahora vamos a ver como es la escritura de estos algoritmos / programas.

Un algoritmo consta de dos componentes: una cabecera de programa y un bloque algoritmo. La cabecera de programa es una acción simple que comienza con la palabra "algoritmo". Esta palabra estará seguida por el nombre asignado al programa completo.

El bloque algoritmo es el resto del programa y consta de dos componentes o secciones: las acciones de declaración y las acciones ejecutables.

Las declaraciones definen o declaran las variables que tengan nombres. Las acciones ejecutables son las acciones que posteriormente deberá realizar la computación cuando el algoritmo convertido en programa se ejecute.

### **Cabecera del programa**

Todos los algoritmos y programas deben comenzar con una cabecera en la que se exprese el identificador o nombre correspondiente con la palabra reservada que señale el lenguaje.

```
Algoritmo sin_titulo
```

```
    <Acciones>
```

```
FinAlgoritmo
```

Donde la palabra sin título debe ser reemplazada por el nombre del algoritmo. Esto se podría ver así:

### **Veamos un ejemplo**

```
1  Algoritmo sin_titulo
2
3
4  FinAlgoritmo
5
```

## ¿Qué elementos posee un programa?

Los elementos de un programa, son básicamente, los componentes que conforman las instrucciones previamente mencionadas, para crear nuestro programa y resolver sus problemas. Estos elementos siempre estarán dentro de un algoritmo.

Los elementos de un programa son: identificadores, variables, constantes, operadores, palabras reservadas.

## Identificadores

Un identificador es un conjunto de caracteres alfanuméricos de cualquier longitud que sirve para identificar las entidades del programa (nombre del programa, nombres de variables, constantes, subprogramas, etc.).

En la mayoría de los lenguajes de programación los identificadores deben constar sólo de letras, números y/o guión bajo (\_), comenzando siempre con una letra y se suelen escribir siempre en minúsculas. Estos tampoco pueden contar de tildes, ni de la letra Ñ, ya que generaría errores.

Otra cosa que es súper importante a la hora de pensar identificadores, es poner nombres claros, por ejemplo, si queremos tener una frase, que el identificador sea frase o si queremos una suma, le pondremos suma.

## Variables y Constantes

Los programas de computadora necesitan información para la resolución de problemas. Esta información puede ser un número, un nombre, etc. Para que podamos guardar esta información en algún lugar y que no esté "suelta", para no perderla o poder acceder a ella cuando lo necesitemos es crucial que guardemos la información en algo llamado, variables y constantes.

Las variables y constantes vendrían a ser como pequeñas cajas, que guardan algo en su interior, en este caso información. Estas van a contar, como previamente habíamos mencionado, con un identificador, un nombre que facilitará distinguir

unas de otras y nos ayudará a saber qué variable o constante es la contiene la información que necesitamos.

Dentro de toda la información que vamos a manejar, a veces, necesitaremos información que no cambie. Tales valores son constantes. De igual forma, existen otros valores que sí necesitaremos que cambien durante la ejecución del programa; esas van a ser nuestras variables.

## Tipos de datos en general

Las variables y constantes, como previamente habíamos mencionado, van a guardar información dependiendo del tipo de dato que le digamos que guarde esa variable. Por ejemplo, si digo que mi variable va a guardar números enteros, significa que el tipo de dato de esa variable es entero.

Los tipos de datos que podemos usar son:

- ✓ Entero: solo números enteros.
- ✓ Real: números con cifras decimales. Para separar decimales se utiliza el punto. Ejemplo: 3.14
- ✓ Carácter: cuando queremos guardar un carácter. Los Caracteres se encierran entre comillas simples. un carácter (unidimensional): 'a', 'A'.
- ✓ Lógico: cuando necesitamos guardar una expresión lógica (verdadero o falso)
- ✓ Cadena: cuando queremos guardar cadenas de caracteres. Las Cadenas se encierran entre comillas dobles. una cadena (multidimensional): "esto es una cadena", "hola mundo"

Los tipos de datos dependen del lenguaje utilizado para programar. Aquí sólo hablaremos de generalidades

## ¿Cómo se crean las Variables?

La definición de variables es relativa de acuerdo al lenguaje de programación que se utiliza. A continuación veremos una forma genérica de hacerlo.

A la hora de crear nuestra variable, vamos a tener que darle un identificador, por lo que usaremos las reglas vistas en el apartado de identificadores, y el tipo de dato que necesitamos que guarde.

Después de la palabra Definir, va el nombre de la variable y por último el tipo de dato de la variable. Normalmente los identificadores de las variables y de las constantes con nombre deben ser declaradas en los programas antes de ser utilizadas. Entonces, la sintaxis de la declaración de una variable suele ser:

```
Definir <nombre_variable> como <tipo_de_dato>
```

### **Veamos un ejemplo**

```
Definir varNumero Como Entero
```

varNumero se convierte en una variable de tipo entero.

## **Tipos de instrucciones**

Las instrucciones —acciones— básicas que se pueden implementar de modo general en un algoritmo y que esencialmente soportan todos los lenguajes son las siguientes:

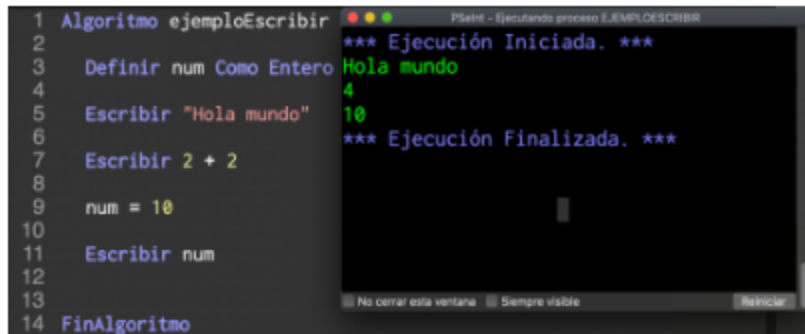
- Instrucciones de inicio/fin

Son utilizadas para delimitar bloques de código. Por ejemplo, Algoritmo y FinAlgoritmo.

- Instrucciones de escritura o salida

Se utilizan para escribir o mostrar mensajes o contenidos de las variables en un dispositivo de salida. La salida puede aparecer en un dispositivo de salida (pantalla, impresora, etc.). La operación de salida se denomina escritura (escribir).

### **Veamos un ejemplo**



```
1 Algoritmo ejemploEscribir
2
3   Definir num Como Entero
4
5   Escribir "Hola mundo"
6
7   Escribir 2 + 2
8
9   num = 10
10
11  Escribir num
12
13
14 FinAlgoritmo
```

\*\*\* Ejecución Iniciada. \*\*\*  
Hola mundo  
4  
10  
\*\*\* Ejecución Finalizada. \*\*\*

En este ejemplo de escribir, vemos que nuestro primer escribir muestra un mensaje o cadena, que va entre comillas dobles, después nuestro segundo escribir muestra el resultado de una suma de dos números y nuestro último escribir, muestra el valor de una variable de tipo entero a la que se le asignó un valor previo.

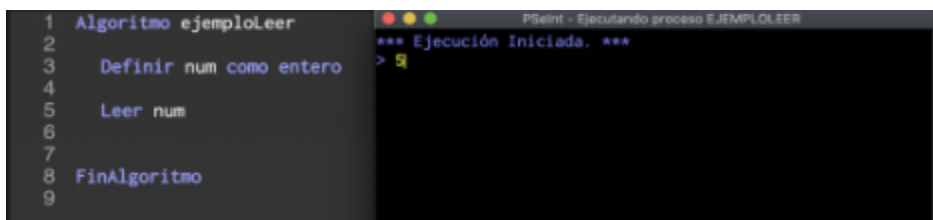
## Instrucciones de lectura

Para que los cálculos que realizan las computadoras sean útiles requieren de la entrada de los datos necesarios para ejecutar las operaciones que posteriormente se convertirán en resultados, es decir, salida.

Las operaciones de entrada permiten leer datos de un dispositivo de entrada y asignarlos a determinadas variables.

Esta entrada se conoce como operación de lectura (leer). Los datos de entrada se introducen al procesador mediante dispositivos de entrada (teclado).

## Veamos un ejemplo



```
1 Algoritmo ejemploLeer
2
3   Definir num como entero
4
5   Leer num
6
7
8 FinAlgoritmo
9
```

\*\*\* Ejecución Iniciada. \*\*\*  
> 5

En este ejemplo definimos una variable de tipo entero llamada num y le asignamos un valor a través de la instrucción Leer.



## ¿Cómo asignamos valores a las variables?

La instrucción de asignación permite almacenar un valor en una variable (previamente definida). Esta es nuestra manera de guardar información en una variable, para utilizar ese valor en otro momento. Se puede realizar con el signo igual:

<variable> = <expresión>

Donde expresión es igual a una expresión matemática o lógica, a una variable o constante.

Al ejecutarse la asignación, primero se evalúa la expresión de la derecha y luego se asigna el resultado a la variable de la izquierda. El tipo de la variable y el de la expresión deben coincidir.

### Veamos un ejemplo

```
1 Algoritmo ejemploAsignacion
2
3   Definir num Como Entero
4
5   num = 4
6
7 FinAlgoritmo
8
```

En este ejemplo estamos definiendo una variable como entero y después asignándole un valor, en este caso el número 4.

## Operadores

Este pseudolenguaje dispone de un conjunto básico de operadores que pueden ser utilizados para la construcción de expresiones más o menos complejas.

- **Operadores Algebraicos**

Los operadores algebraicos o también conocidos como operadores aritméticos. Realizan operaciones aritméticas básicas: suma, resta, multiplicación, división,

potenciación y módulo para datos de tipo numérico tanto enteros como reales. Estas operaciones son binarias porque admiten dos operandos.

Operadores Algebraicos	Significado	Resultado
+	Suma	suma = 2 + 2
-	Resta	resta = 10 - 4
*	Multiplicación	multiplicación = 10 * 2
/	División	división = 9 / 3
^	Potenciación	potencia = 10 ^ 2
% o MOD	Módulo (resto de la división entera)	resto = 4 % 2

## Estructura de Control

Las Estructuras de Control determinan el orden en que deben ejecutarse las instrucciones de un algoritmo, es decir, si serán recorridas una después de la otra (estructuras secuenciales), si habrá que tomar decisiones sobre si ejecutar o no alguna acción (estructuras selectivas o de decisión) o si habrá que realizar repeticiones (estructuras repetitivas). Esto significa que una estructura de control permite que se realicen unas instrucciones y omitir otras, de acuerdo a la evaluación de una condición.

Esto hace que las estructuras de control se puedan dividir en tres:

- **Estructuras secuenciales**

Es la estructura en donde una acción (instrucción) sigue a otra de manera secuencial. Las tareas se dan de tal forma que la salida de una es la entrada de la que sigue y así en lo sucesivo hasta cumplir con todo el proceso. Esta estructura de control es la más simple, permite que las instrucciones que la constituyen se ejecuten una tras otra en el orden en que se listan.

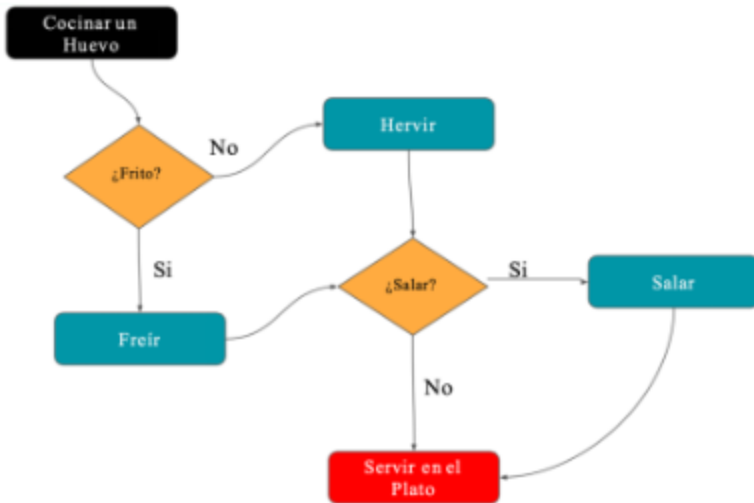


- **Estructuras selectivas o de decisión**

Estas estructuras de control son de gran utilidad para cuando el algoritmo a desarrollar requiera una descripción más complicada que una lista sencilla de instrucciones. Este es el caso cuando existe un número de posibles alternativas que resultan de la evaluación de una determinada condición. Este tipo de estructuras son utilizadas para tomar decisiones lógicas, es por esto que también se denominan estructuras de decisión o selectivas.

En estas estructuras, se realiza una evaluación de una condición y de acuerdo al resultado, el algoritmo realiza una determinada acción. Las condiciones son especificadas utilizando expresiones lógicas.

Una estructura selectiva, con varias condiciones, sería el ejemplo que usamos en la primera parte de Cocinar un Huevo. Las preguntas serían las condiciones a evaluar y de acuerdo a ese resultado realiza una o u otra acción.



- **Estructuras repetitivas**

Son aquellas que repiten las acciones determinada cantidad de veces en función de una condición o en función a una cantidad establecida por el programador.

### ¿Qué es una condición?

En programación, una condición es toda sentencia de la cual se puede determinar su verdad (true) o falsedad (false). En su gran mayoría, son comparaciones. Por ejemplo,  $4 > 5$ , esta sentencia es una condición porque tiene resultado verdadero o falso, en este caso falso porque 4 no es mayor a 5. En cambio, la siguiente sentencia, Escribir "EggEducacion", no es condición puesto que no hay para comparar, no se puede determinar verdad o falsedad.

Por lo que una condición sirve para discernir entre una opción u otra, y en el proceso mental normalmente se manifiesta con un "Si"; por ejemplo: Si (va a llover), coge el paraguas.

Para determinar condiciones, precisamos utilizar Operadores.

### ¿Qué son los operadores?

Las condiciones que usaremos en las estructuras selectivas y el resto de nuestras estructuras de control se realizan con la ayuda de los operadores relacionales y lógicos.

Los operadores relacionales son símbolos que se usan para comparar dos valores. Si el resultado de la comparación es correcto la expresión considerada es verdadera, en caso contrario es falsa.

Operadores Relacionales	Significado	Ejemplo
>	Mayor que	3 > 2 // verdadero
<	Menor que	1 < 5 // verdadero
==	Igual que	4 == 4 // verdadero
>=	Mayor o igual que	4 >= 5 // falso
<=	Menor o igual que	'a' <= 'b' // verdadero
<> o !=	Distinto que	10 <> 8 // verdadero

## Operadores Lógicos

Estos se utilizan cuando necesitamos las expresiones lógicas con múltiples variantes y nos proporcionan un resultado a partir de que se cumpla o no una cierta condición, estos producen un resultado lógico, y sus operadores son también valores lógicos o asimilables a ellos.

Operadores Lógicos	Significado	Ejemplo
Y	Conjunción	(2 < 4 Y 3 > 5) // falso
O	Disyunción	(7 <= 8 O 10 >= 9) // verdadero
NO / no	Negación	no(1 == 1) // falso

## Operador Y

Devuelve un valor lógico verdadero si ambas expresiones son verdaderas. En caso contrario el resultado es falso.

## Operador O

Este operador devuelve verdadero si alguna de las expresiones es verdadera. En caso contrario devuelve “falso”.

## Operador NO

Este operador cambia la devolución de una expresión, al caso contrario. Si es verdadero lo hace falso y si es falso lo hace verdadero.

A la hora de trabajar con operadores lógicos, para saber si una expresión lógica nos devuelve como resultado Verdadero o Falso, debemos observar la siguiente tabla de la verdad:

Conjunción				Disyunción			
A	Operador	B	Resultado	A	Operador	B	Resultado
V	Y	V	V	V	O	V	V
V	Y	F	F	V	O	F	V
F	Y	V	F	F	O	V	V
F	Y	F	F	F	O	F	F

Negación			
A	Resultado	B	Resultado
no(V)	F	no(F)	V

## Veamos un ejemplo

Vamos a mostrar ejemplos de condiciones tanto con operadores relacionales, como con lógicos

$x=y$ , significa “si x es igual a y”

$x>y$ , significa “si x es mayor que y”

$x<y$ , significa “si x es menor que y”

$x!=y$ , significa “si x es distinto de y”

$(x=j) \text{ Y } (x=z)$ , significa "si x es igual a j Y x igual a z"

$(x=y) \text{ O } (x=z)$ , significa "si x es igual a j O x igual a z"

## Estructuras selectivas

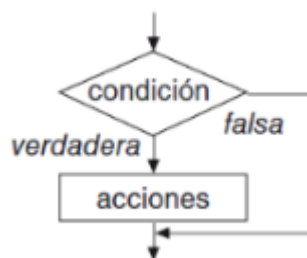
Entendemos que las estructuras selectivas son utilizadas para tomar decisiones lógicas, es por esto que también se denominan estructuras de decisión o selectivas. Pero, ¿cuáles son las estructuras selectivas?

Las estructuras selectivas/alternativas pueden ser:

- Simples: Si
- Doble: Si- SiNo
- Múltiples: Según – Si Anidado

## Condición Simple

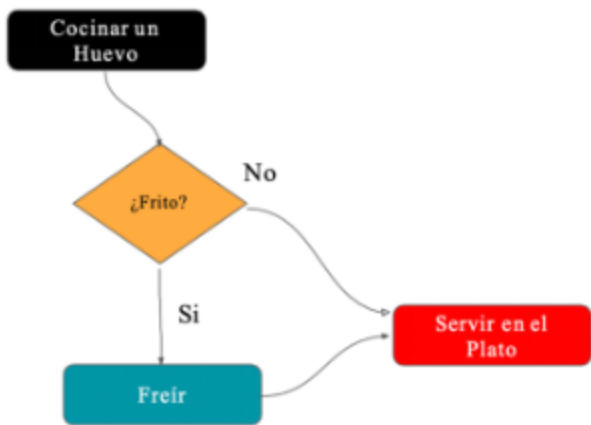
La estructura alternativa simple si-entonces lleva a cabo una acción siempre y cuando se cumpla una determinada condición.



La selección si-entonces evalúa la condición y luego:

- Si la condición es verdadera, ejecuta el bloque de acciones
- Si la condición es falsa, no ejecuta otra opción.

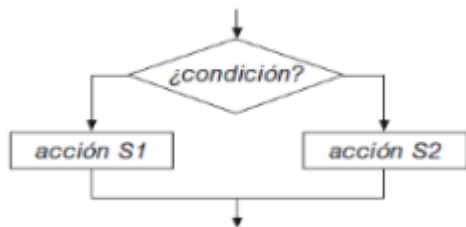
## Veamos un ejemplo



Si fuera cocinar un huevo, tenemos solo la opción de freírlo y si no lo queremos frito, se va servir crudo en el plato. Esto es una Condición Simple.

### Condición Doble

La estructura anterior es muy limitada y normalmente se necesitará una estructura que permita elegir entre dos opciones o alternativas posibles, en función del cumplimiento o no de una determinada condición. Si la condición es verdadera, se ejecuta la acción S1 y, si es falsa, se ejecuta la acción S2.

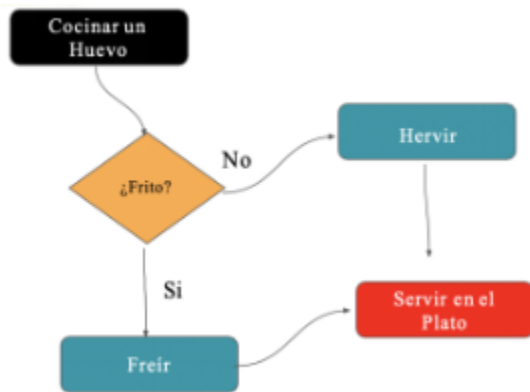


La selección si-entonces-sino evalúa la condición y luego:

- Si la condición es verdadera, ejecuta el bloque de acciones
- Si la condición es falsa, ejecuta el bloque de acciones 2.



## Veamos un ejemplo

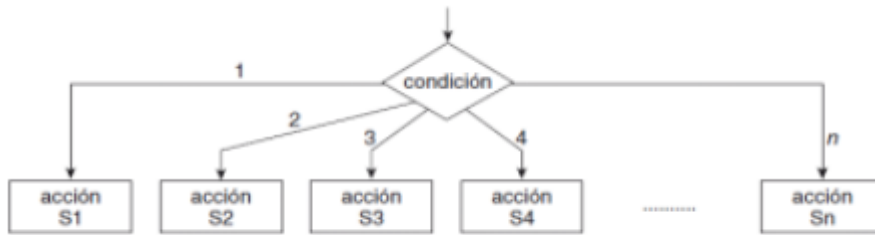


En este caso, si fuera cocinar un huevo, tenemos opción de freírlo y si no lo queremos frito, tendremos la opción de hervirlo. Esto es una Condición Doble.

## **Condición Múltiple**

Muchas veces vamos a tener más de dos alternativas para elegir, o una variable que puede tomar varios valores. Para solucionar esto, usamos la condición múltiple. En esta estructura, se evalúa una condición o expresión que puede tomar  $n$  valores. Según el valor que la expresión tenga en cada momento se ejecutan las acciones correspondientes al valor.

La estructura de decisión múltiple evaluará una expresión que podrá tomar  $n$  valores distintos, 1, 2, 3, 4, ...,  $n$ . Según el valor que elija en la condición, se realizará una de las  $n$  acciones, o lo que es igual, el flujo del algoritmo seguirá un determinado camino entre los  $n$  posibles. Por ejemplo, si tenemos un sistema de notas, donde 6 es desaprobado, 7 es aprobado, 9 es sobresaliente y 10 es excelente. Al tener un valor que puede dar distintas alternativas, usamos la condición múltiple.



Este problema, se podría resolver por estructuras alternativas simples o dobles, anidadas o en cascada; sin embargo, este método si el número de alternativas es grande puede plantear serios problemas de escritura del algoritmo y naturalmente de legibilidad.