



# Introducción al Pensamiento Computacional - Semana 2



© Todos los derechos reservados Universidad Rafael Landívar URL.

---

## DESEMPEÑOS ESPERADOS

### ☰ Desempeños esperados

---

## DESARROLLO DE CONOCIMIENTOS

### ☰ Enfoque

### ☰ Pensamiento lógico

### ☰ Pensamiento algorítmico

### ☰ Recursos complementarios

---

## APLICANDO LO APRENDIDO

### ☰ Actividad 1

### ☰ Actividad 2

### ☰ Actividad 3

### ☰ Actividad 4

### ☰ Actividad 5

---

## DE LA TEORÍA A LA PRÁCTICA Y REFLEXIÓN

### ☰ Recursos

---

## CRITERIOS DE EVALUACIÓN

### ☰ Rúbrica de evaluación

### ☰ Diario de experiencias de laboratorio

---

## FUENTES DE REFERENCIAS

 Referencias

CRÉDITOS

---

 Créditos

## Desempeños esperados

---



### El estudiante:

1

Identifica la importancia de la lógica en el pensamiento computacional.

2

Valorar la diferencia entre el razonamiento deductivo y el razonamiento inductivo.

3

Comprende la lógica booleana y su importancia en la computación.

4

Comprender la importancia de utilizar la notación de lógica matemática en lugar de lenguaje neutral.

5

Identificar las propiedades de los algoritmos: secuencia, iteración y selección.

6

Comprender la importancia del estado en los algoritmos.

7

Detectar errores comunes en el pensamiento lógico y algorítmico y aplicar estrategias para evitarlos.

## Enfoque

---



La lógica y los algoritmos son esenciales en el pensamiento computacional. Afortunadamente, los seres humanos tienen una concepción innata de la lógica y los algoritmos. Sin embargo, es necesario conocer las reglas, procedimientos y definiciones, las cuales son precisas y sistemáticas. Esto significa que no es suficiente la intuición para aplicar estos temas, de lo contrario se pueden cometer errores.

El estudio de la lógica y los algoritmos es muy amplio, en este módulo se abordarán los elementos esenciales y relevantes para establecer el hábito de pensar lógica y algorítmicamente para aplicarlos a la resolución de problemas. Es importante la práctica constante para desarrollar las habilidades y destrezas de pensamiento.



## Pensamiento lógico

---

### Lógica



La **lógica** es un sistema utilizado para distinguir entre los argumentos correctos e incorrectos. Un **argumento** es una idea filosófica, encadenada a un **razonamiento** para llegar a una **conclusión**.

La lógica incluye un conjunto de principios, que al aplicarlos a los argumentos, permiten demostrar lo que es verdad. Analicemos un ejemplo para el cual no se requiere un entrenamiento especial:

- 1 Sócrates es un humano.
- 2 Todos los humanos son mortales.
- 3 Entonces, Sócrates es mortal.

No es necesario tener grados avanzados en filosofía para comprender estos temas, sin embargo, no siempre la intuición conduce a resultados correctos, lo que puede influir en realizar conclusiones erróneas. Al utilizar las computadoras para automatizar el razonamiento, es importante que aprendamos la lógica, previo a escribir una solución computacional.



La aplicación de la lógica es una forma de desarrollar y probar una hipótesis. Al aplicar la lógica, se asume que ya se tienen algunos presaberes y se dan por ciertos, lo que permite utilizar estos conocimientos para llegar a futuras conclusiones.

En un argumento lógico, cada cosa que se ya se sabe o asume se llama premisa. La premisa es un enunciado que se realiza y debe ser evaluado para responder si es "**falso**" o "**verdadero**".

La premisa entonces es un valor de verdad.

## Ejemplos

Sócrates es un humano.

Es una premisa, porque puede

**ser contestada como  
verdadero o falso.**

Todos los humanos son  
mortales.

**Es una premisa, porque puede  
ser contestada como  
verdadero o falso.**

¡Buena suerte!

**No es una premisa, porque no  
puede ser contestado como  
falso o verdadero.**

¿Qué hora es?

**No es una premisa, porque no  
puede ser contestado como  
falso o verdadero.**

Una vez se definen las premisas, el siguiente paso es analizarlas y reaccionar con una conclusión. Mucha de la "magia" radica en el proceso de concluir.



# Argumentos deductivos versus argumentos inductivos

Algunos argumentos son más fuertes que otros. Se pueden clasificar según su nivel de certeza. Las clasificaciones más conocidas son deductivos e inductivos.

Un **argumento deductivo** es la forma más fuerte de razonar porque la conclusión se deriva necesariamente de las premisas. Un argumento deductivo puede fallar de dos maneras.

## 1. Cuando una de sus premisas es falsa.

Por ejemplo:

- 1.1 Lady es un perro.
- 1.2 Todos los perros son de color cafés.
- 1.3 Entonces, Lady es de color café.

La premisa 1.2 es falsa. No todos los perros son de color café. Aunque el argumento sigue la misma forma del ejemplo de Sócrates, en este caso falla, porque al menos una de sus premisas es falsa. Cualquier argumento con premisas falsas, falla.

## 2. Cuando la conclusión no necesariamente se deriva de las premisas.

Por ejemplo:

- 2.1 Todas las pelotas de tenis son redondas.
- 2.2 La Tierra es redonda.
- 2.3 Entonces, la Tierra es una pelota de tenis.

Este argumento falla por una lógica defectuosa. Sí, es verdadero que todas las pelotas de tenis son redondas, pero también otras cosas lo son. A un argumento fallido se le conoce como **falacia**.

Los argumentos deductivos son raros. Se pueden encontrar en escenarios donde el conocimiento con el que se está tratando es limpio. En la "vida real", el conocimiento es más desordenado o provisional. Los desafíos de la vida real tienen más matices y para ello existe el **argumento inductivo**, el cual se enfoca en **probabilidades**, más que en reglas rígidas.

Las premisas de un argumento inductivo no son verdaderas absolutamente. Tienen cierto nivel de confianza en ellas. La forma de un argumento no garantiza que la conclusión sea verdadera, pero probablemente resulta en una conclusión confiable. Por ejemplo:

- 1 Una bolsa contiene 99 pelotas rojas y 1 pelota negra.
- 2 100 personas toman, cada una, una pelota de la bolsa.
- 3 Alba es una de esas 100 personas.
- 4 Entonces, Alba probablemente obtendrá una pelota roja.

Estos aspectos del razonamiento son importantes en pensamiento computacional porque las computadoras están involucradas. La respuesta que da una computadora es tan confiable como su razonamiento y las computadoras automatizan el razonamiento humano. Es la responsabilidad del pensador computacional:

- 1 Que el razonamiento sea válido.
- 2 Proveer una entrada confiable.
- 3 Saber interpretar la conclusión que la computadora reporta, es decir, si es una verdad incuestionable (razonamiento deductivo) o una verdad probable (razonamiento inductivo).



A pesar de que mucho del razonamiento es inductivo, las computadoras tienen una naturaleza binaria, las cuales las hace más aptas para procesar dicotomías. Para darle instrucciones a una computadora sobre cómo realizar decisiones lógicas, se necesita un sistema de lógica que se adecúe a su naturaleza. Para ello se utiliza la **lógica booleana**.

La lógica booleana es una forma de lógica que trata a los enunciados con dos posibles valores: verdadero o falso (el más usual). También se pueden utilizar otros valores según el contexto: 1 ó 0, on u off, blanco o negro.

## Proposiciones

Los enunciados en lógica booleana también son conocidos como proposiciones, los cuales tienen propiedades básicas.

1.

2.

3.

Una proposición solo puede tener un valor a la vez. Por ejemplo: Una proposición NO PUEDE ser falsa y verdadera a la vez. No hay forma de expresar niveles de certeza. Verdadero es verdadero y falso es falso.

1.

2.

3.

Una proposición debe ser clara y su significado debe evita la ambigüedad. Por ejemplo: si un enunciado dice "**Está viajando rápido**", puede ser evaluado como falso o verdadero. Sin embargo, existe ambigüedad, ya que si quien está viajando es un carro, una velocidad de 160 km/h puede considerarse rápida. Pero si quien está viajando es una nave espacial, ese mismo valor es definitivamente lento.

1.

2.

3.

Es posible combinar proposiciones para realizar otras más completas (proposiciones compuestas). Por ejemplo: "Jenny está usando la falda y la falda es roja". Esto puede ser útil, ya que muchas veces necesitamos evaluar varias proposiciones a la vez. Note que las

proposiciones compuestas se conectan con **operadores lógicos**.

## Operadores lógicos

Imagínese diciendo: "Si el clima está soleado y estoy de vacaciones, entonces me iré a recostar al jardín". Pues acaba de utilizar operadores lógicos. Este enunciado contiene dos proposiciones, que sirven de condición para decidir si recostarse o no en la grama:

1

Si el clima está soleado.

2

Si estoy de vacaciones.

Si los dos son verdaderos, se puede acostar en el jardín. Si una de las proposiciones es falsa (si el clima es terrible o tiene que ir a trabajar/estudiar), entonces asolearse en el jardín no es una opción. Todo gracias al operador **Y**, el cual demanda que ambas proposiciones sean verdaderas.

Existen muchos operadores lógicos, vale la pena mirar los más importantes en detalle, ya que se utilizan de manera informal cuando hablamos. Esto implica que la lógica se encuentra en la comprensión cotidiana. A continuación, ejemplos ilustrativos.

### Y (AND)

**Y (AND):** el nombre técnico es **conjunción**. Encadena proposiciones juntas de manera que **todas ellas deben ser verdaderas** para que la conclusión sea verdadera. Si una de las proposiciones es falsa, la conclusión también será falsa. Por ejemplo:

1

Al menos un cuadro del tablero está vacío ( $P$ ). A

2

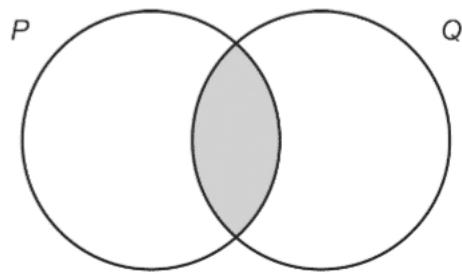
Ningún jugador ha finalizado una fila (Q).

3

Entonces, el juego aún está en proceso.

Se puede expresar como:

En la siguiente figura se observa el diagrama de Venn del operador Y (AND).



**Si** (al menos un cuadrado del tablero está vacío) **Y** (ningún jugador ha finalizado una fila),  
**Entonces**, el juego aún está en proceso.

En las siguientes tablas, se puede observar los valores del operador Y (AND), con valores de verdadero/falso y 1/0 respectivamente.

<b>P</b>	<b>Q</b>	<b>P AND Q</b>
True	True	True
True	False	False
False	True	False
False	False	False

$P$	$Q$	$P \text{ AND } Q$
0	0	0
0	1	0
1	0	0
1	1	1



O (OR)

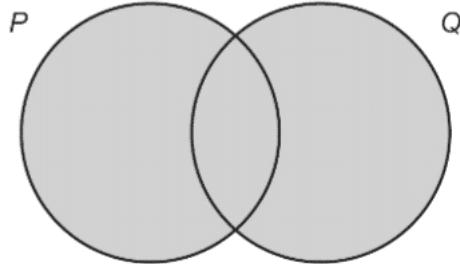
**O (OR):** el nombre técnico es **disyunción**. Este operador encadena las proposiciones juntas de manera que **al menos una de ellas sea verdadera** para que la conclusión también sea verdadera. La única forma que la conclusión sea falsa es que todas las proposiciones sean falsas. Por ejemplo:

**Si** (el jugador 1 alcanza una fila) **O** (el jugador 2 alcanza una fila), **Entonces**, el juego ha terminado.

En este caso solo una condición necesita ser verdadera para que el juego termine. Incluso, permite que ambos jugadores terminen simultáneamente. O también puede ser usado en los casos donde las dos condiciones son verdaderas al mismo tiempo. Por ejemplo:

**Si** (el jugador alcanza una fila (P)) **O** (todos los cuadros se ocupan (Q)), **Entonces**, el juego ha terminado.

En la siguiente figura se observa el diagrama de Venn del operador O (OR).



En la siguiente tabla, se puede observar los valores del operador O (OR).

<b>P</b>	<b>Q</b>	<b>P AND Q</b>
True	True	True
True	False	False
False	True	False
False	False	False



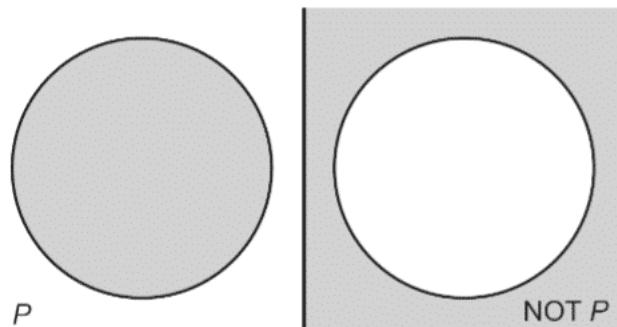
NO (NOT)

**NO (NOT):** el nombre técnico es **negación**. Este operador no encadena proposiciones por sí mismo, sino modifica una sola proposición. Específicamente, cambia invierte el valor de verdad. Algunas veces, negar una proposición puede ser más fácil para expresar una cadena de razonamiento. Por ejemplo:

**Si (un cuadro no está ocupado ( $\text{NOT } P$ ))**

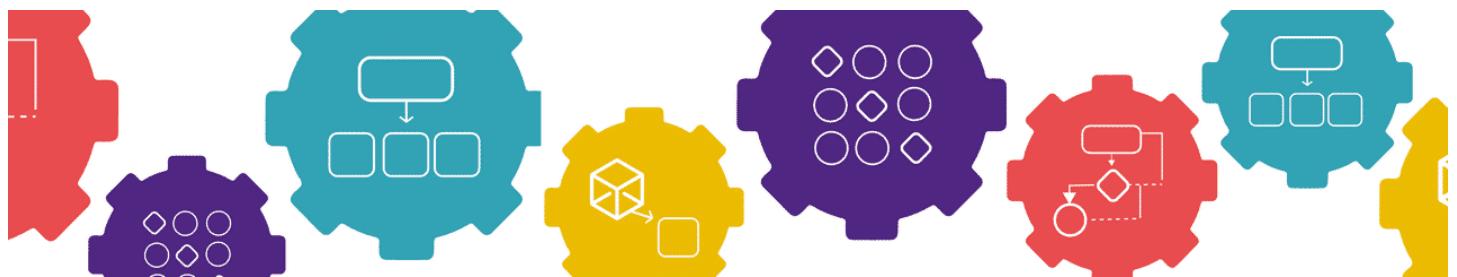
**Entonces, el jugador puede agregar su ficha al cuadro.**

En la siguiente figura se observa el diagrama de Venn del operador NO (NOT).



En la siguiente tabla, se puede observar los valores del operador NO (NOT).

<b><math>P</math></b>	<b><math>\text{NOT } P</math></b>
True	False
False	True



## Pensamiento algorítmico



Nos movemos ahora de pensamiento lógico a pensamiento algorítmico. En esta sección se introducen las propiedades de los algoritmos (secuencia, iteración y selección), así como el uso del estado.

### Algoritmos

La lógica y los algoritmos no son lo mismo. Sin embargo, los algoritmos se construyen con base a la lógica, porque como parte de su trabajo, realizan decisiones lógicas. Otra parte de su trabajo es integrar las decisiones.

La lógica nos provee reglas que permiten razonar sobre un aspecto del mundo. Ese mundo no tiene por qué ser estático. La lógica puede tratar con situaciones que son dinámicas y cambiantes continuamente.

Al progresar en el estudio del pensamiento computacional, queremos construir sistemas basados en reglas, así que sólo hacerlo con lógica no es suficiente. Necesitamos un mecanismo que integre las reglas y ejecute las acciones basadas en la evaluación de dichas reglas. Ese mecanismo son los algoritmos, son el poder detrás de los sistemas computacionales.

*Algoritmo: es una secuencia de pasos definidos claramente que describen una ruta para seguir un conjunto finito de instrucciones no ambiguas con inicio y finales claramente establecidos.*

Los algoritmos son formas de especificar tareas de múltiples pasos y son útiles cuando queremos explicar a un tercero (humano o máquina) cómo llevar a cabo los pasos con extrema precisión. Al igual que con la lógica, las personas ya poseen una comprensión intuitiva de los algoritmos. Sin embargo, hay principios que deben aprenderse, ya que los **algoritmos precisos** son la base de las soluciones basadas en computadoras.

## Intuición versus precisión

En la vida diaria podemos ver ejemplos de algoritmos, como:

### Una receta de cocina



## Receta Arroz con leche

### INGREDIENTES

- 200 g de arroz
- 1 litro de leche
- 1 vaina de vainilla abierta y raspada
- 150 g de azúcar
- 1/2 cucharadita de canela en polvo

### INSTRUCCIONES

1. Calienta el arroz con la mitad de la leche en una olla mediana a fuego medio.
2. Cuando el arroz haya absorbido casi todo el líquido, agrega el resto junto con la vainilla, el azúcar y la canela.
3. Continúa cociendo por 8 minutos más o hasta que el arroz esté suave. Posteriormente sirve y acompaña con un poco de canela en polvo encima

Tiempo:  
1h

Cocina:  
mexicana

Porciones:  
4 porciones

Tomado de: <https://www.cocinafacil.com.mx/recetas-de-comida/receta/arroz-con-leche-casero/>

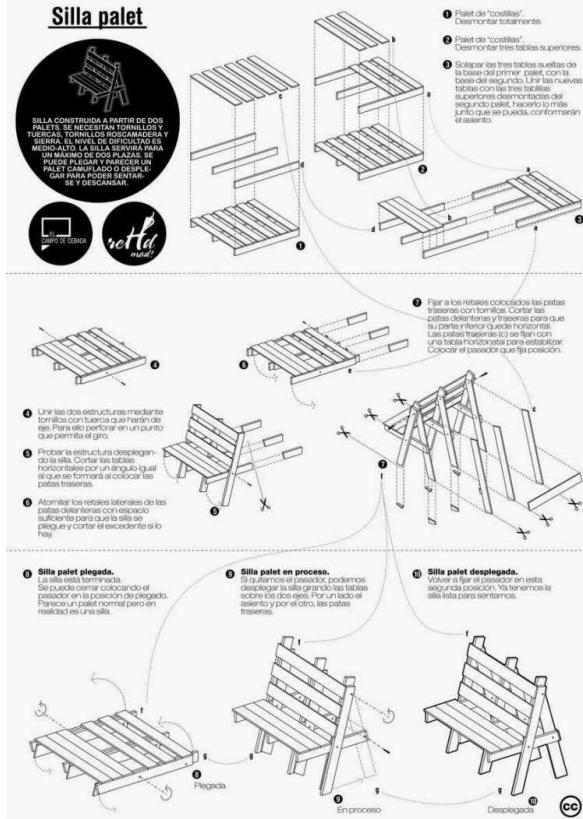
**Un juego de la búsqueda de un tesoro**



Tomado de: <https://www.mapainteractivo.net/fotos/mapa-del-tesoro.html>

---

### Instrucciones de cómo armar un mueble



Tomado de: [https://bricolaje.facilisimo.com/d/instrucciones-para-realizar-una-silla-o-mesa-con-palets\\_1936321.html](https://bricolaje.facilisimo.com/d/instrucciones-para-realizar-una-silla-o-mesa-con-palets_1936321.html)

En estos ejemplos podemos ver la forma de hacer explícito un proceso para comunicarlo a alguien más, de forma que pueda seguir los mismos pasos. Estas son sólo analogías, sin embargo, los algoritmos computacionales, además, pueden realizar operaciones en los datos (más allá que hornear un pastel o ensamblar una mesita de centro).

Para poder darle instrucciones a las computadoras, debemos utilizar ciertos símbolos y lenguajes, que las estructuran de manera que las computadoras las puedan entender y procesar. Estas instrucciones deben ser precisas y deben evitar la ambigüedad.

## Definición de algoritmos

## **Definición de algoritmos**

La definición de algoritmos implica las siguientes propiedades.

## Paso 1

### Colección de pasos individuales



Un algoritmo es una colección de pasos individuales. El ejemplo de la receta de cocina encaja perfectamente: "precaliente el horno a 180 grados Celsius" o "agregue dos cucharaditas de azúcar al tazón".

## Paso 2

### Definición

Cada paso debe ser definido de manera precisa. Cada paso del algoritmo debe tener solo un significado, de lo contrario puede ser ambiguo. Siguiendo con el ejemplo de la receta de cocina, se evita la utilización de "algo de azúcar" o "cocínelo por un rato", en su lugar se dan medidas y datos precisos.

### Paso 3

## Secuencia

Los algoritmos son secuenciales. Los pasos para hacer que el proceso sea llevado a cabo en un orden específico. Fallar en el orden implica que el resultado será incorrecto. Continuando con la analogía de la receta de cocina: rodajear una cebolla o freírla son pasos diferentes y separados. Rodajear antes de freír tiene un resultado diferente a freír y luego rodajear. Se debe respetar la secuencia cuando se ejecuta un algoritmo para que el resultado sea significativo.

#### Paso 4

### Estado

El estado significa mantener el registro/seguimiento de los valores de todos los componentes del algoritmo. El estado de las cosas va cambiando conforme se ejecuta el algoritmo. Con ejemplo de la receta: al inicio se tienen los ingredientes por separado, pero van cambiando de forma, de nivel de cocción, preparación y presentación, a medida que avanza la ejecución de la receta.

## Estado

La configuración actual de toda la información de la cual se lleva registro por un programa a cada instante del tiempo.

Cada vez que la computadora ejecuta una operación, **"olvida"** lo que pasó en la operación anterior. Si queremos que la computadora **"recuerde"** cierto resultado para usos posteriores, se le debe dar la instrucción de hacerlo. Para ello, se le da la instrucción de guardar los datos en **variables**.

Aunque se comparta el mismo nombre en matemáticas y en programación, las variables no significan lo mismo en estas dos áreas. En programación, una variable es como una notita adhesiva (post-it) que se usa para colocar información importante que la computadora deba considerar o recordar.

## Controlar la ejecución de un algoritmo

Escuche esta canción y analícela.



En este ejemplo de iteración, se pueden observar dos cosas.

¿De qué trata? ¿Cómo haría un algoritmo que describa la letra de la canción?

Sería una escritura que contenga esta estructura:

99 bottles of beer on the wall,

99 bottles of beer.

Take one down, pass it around,

98 bottles of beer on the wall.

99 bottles of beer on the wall,

99 bottles of beer.

...

Y se seguiría repitiendo hasta llegar al último verso:

```
1 bottle of beer on the wall,  
1 bottle of beer.  
Take one down, pass it around,  
No more bottles of beer on the wall.
```

¡Ahórrese el escribir todo eso! Aún con el copy + paste. En su lugar escriba un solo verso de la canción y dele las instrucciones a la computadora para que lo repita, con ayuda de variables. A esto se le llama **ITERACIÓN**. El resultado quedaría así:

```
X se refiere al número de bottles  
Al inicio el valor de X es 99  
Ahora cante el verso siguiente hasta que X sea mayor a 0:
```

```
X bottles of beer on the wall,  
X bottles of beer.  
Take one down, pass it around,  
X-1 bottles of beer on the wall.
```

En este ejemplo de iteración, se pueden observar dos cosas:

- Cómo una variable se utiliza para controlar la ejecución de un algoritmo.
- Se debe especificar en qué condiciones sigue repitiéndose el ciclo de los versos y cuándo debe terminar.

Este segundo aspecto, nos conduce al otro método para controlar la ejecución de un algoritmo: la **SELECCIÓN**.

La selección o condicional, es una forma de validar el valor actual de una variable y realizar una decisión con esa base. Las condiciones se pueden utilizar en cualquier parte de un algoritmo, no solo para controlar ciclos. El objetivo de las selecciones es crear un punto donde la computadora deba decidir entre realizar una tarea o no realizarla.

## Ejemplo de algoritmo

Este ejemplo ilustra todo lo mencionado anteriormente. Se trata de un algoritmo para controlar un juego. Se escribió en **pseudocódigo**, que es más informal y está dirigido a un público humano, más que a una computadora. Sin embargo, se acerca a las convenciones de los lenguajes

de programación, de esta forma puede ser traducido fácilmente.

- 1 Empieza el juego
- 2 Empieza el ciclo
  - 3 Pídale al jugador que escoja un cuadro.
  - 4 Si elige un cuadrado que no está ocupado, entonces
    - Coloque el símbolo del jugador en el cuadro
  - 5 Revise el tablero para ver si se ha completado una fila
  - 6 Si una fila se ha completado, entonces
    - El juego se ha ganado
  - 7 Si una fila se ha completado y no hay cuadros disponibles, entonces
    - El juego es completado
  - 8 Cambie al otro jugador
- 9 Salir del ciclo si el juego se ha ganado o si el juego es completado
- 10 Despliegue el mensaje "El juego ha terminado"

## Analicemos el algoritmo:

Cuando el algoritmo es ejecutado y la computadora va una línea a la vez, es un ejemplo de **secuencia**.

- 1 Empieza el juego
- 2 Empieza el ciclo
  - 3 Pídale al jugador que escoja un cuadro.
  - 4 Si elige un cuadrado que no está ocupado, entonces
    - Coloque el símbolo del jugador en el cuadro
  - 5 Revise el tablero para ver si se ha completado una fila
  - 6 Si una fila se ha completado, entonces
    - El juego se ha ganado
  - 7 Si una fila se ha completado y no hay cuadros disponibles, entonces
    - El juego es completado
  - 8 Cambie al otro jugador
- 9 Salir del ciclo si el juego se ha ganado o si el juego es completado
- 10 Despliegue el mensaje "El juego ha terminado"

1 Empleza el juego

2 Empleza el ciclo

3 Pídale al jugador que escoja un cuadro.

4 Si elige un cuadrado que no está ocupado, entonces

Coloque el símbolo del jugador en el cuadro

5 Revise el tablero para ver si se ha completado una fila

6 Si una fila se ha completado, entonces

El juego se ha ganado

7 Si una fila se ha completado y no hay cuadros disponibles, entonces

El juego es completado

8 Cambie al otro jugador

9 Salir del ciclo si el juego se ha ganado o si el juego es completado

10 Despliegue el mensaje "El juego ha terminado"

1.

Línea 1: Inicializa la **variable** juego. El valor asignado es iniciado (**estado**).

- 1 Empleza el juego
- 2 Empieza el ciclo
  - 3 Pídale al jugador que escoja un cuadro.
  - 4 Si elige un cuadrado que no está ocupado, entonces
    - Coloque el símbolo del jugador en el cuadro
  - 5 Revise el tablero para ver si se ha completado una fila
  - 6 Si una fila se ha completado, entonces
    - El juego se ha ganado
  - 7 Si una fila se ha completado y no hay cuadros disponibles, entonces
    - El juego es completado
  - 8 Cambie al otro jugador
  - 9 Salir del ciclo si el juego se ha ganado o si el juego es completado
  - 10 Despliegue el mensaje “El juego ha terminado”

2.

Línea 2: Inicia el ciclo (**iteración**). Las líneas que están indentadas (anidadas) se colocan así para expresar que están dentro del ciclo.

- 1 Empieza el juego
- 2 Empieza el ciclo
- 3 Pídale al jugador que escoja un cuadro.
- 4 Si elige un cuadrado que no está ocupado, entonces
  - Coloque el símbolo del jugador en el cuadro
  - 5 Revise el tablero para ver si se ha completado una fila
  - 6 Si una fila se ha completado, entonces
    - El juego se ha ganado
  - 7 Si una fila se ha completado y no hay cuadros disponibles, entonces
    - El juego es completado
  - 8 Cambie al otro jugador
- 9 Salir del ciclo si el juego se ha ganado o si el juego es completado
- 10 Despliegue el mensaje “El juego ha terminado”

3.

Línea 3: pide una entrada del jugador y la respuesta se almacena.

- 1 Empleza el juego
- 2 Empleza el ciclo
- 3 Pídale al jugador que escoja un cuadro.
- 4 Si elige un cuadrado que no está ocupado, entonces
  - Coloque el símbolo del jugador en el cuadro
- 5 Revise el tablero para ver si se ha completado una fila
- 6 Si una fila se ha completado, entonces
  - El juego se ha ganado
- 7 Si una fila se ha completado y no hay cuadros disponibles, entonces
  - El juego es completado
- 8 Cambie al otro jugador
- 9 Salir del ciclo si el juego se ha ganado o si el juego es completado
- 10 Despliegue el mensaje "El juego ha terminado"

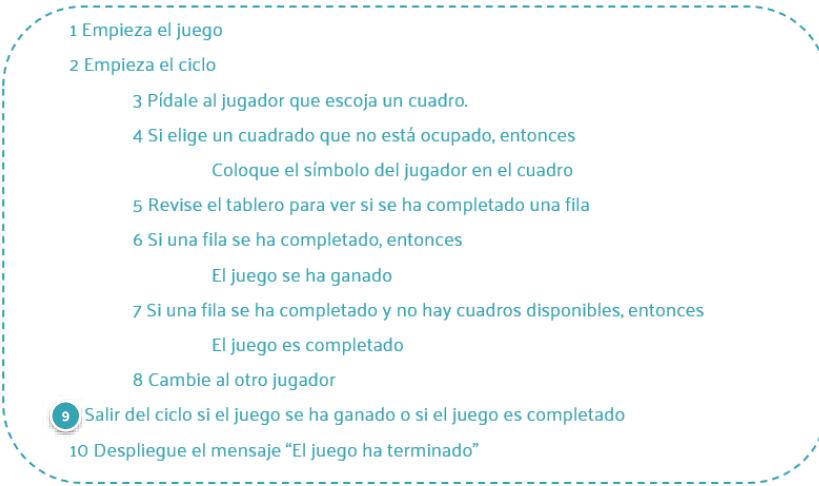
4.

Línea 4: realiza una **selección** basada en la decisión del jugador.

- 1 Empieza el juego
- 2 Empieza el ciclo
  - 3 Pídale al jugador que escoja un cuadro.
  - 4 Si elige un cuadrado que no está ocupado, entonces
    - Coloque el símbolo del jugador en el cuadro
  - 5 Revise el tablero para ver si se ha completado una fila
  - 6 Si una fila se ha completado, entonces
    - + El juego se ha ganado
  - 7 Si una fila se ha completado y no hay cuadros disponibles, entonces
    - El juego es completado
  - 8 Cambie al otro jugador
- 9 Salir del ciclo si el juego se ha ganado o si el juego es completado
- 10 Despliegue el mensaje “El juego ha terminado”

## 6 y 7.

Líneas 6 y 7: ambas selecciones alteran el valor de la variable juego, bajo ciertas condiciones.



## 9.

Línea 9: es el fin del ciclo. Se realiza una decisión para saber si se debe ingresar al ciclo otra vez según la condición a este punto. Hay dos posibilidades:

- El juego no ha sido alterado, en este caso la ejecución se mueve a la línea 3.
- En cierto punto el juego está ganado o completado. En cualquiera de los casos, el ciclo termina y la ejecución continúa en la línea 10. Esa línea despliega el mensaje de "Juego terminado".

### Importante recordar:

- Una computadora realizará exactamente lo que se le diga. Si se le dice algo imposible colapsará.
- Una computadora no tiene una inteligencia innata en sí misma. No hará algo que no se le haya indicado.
- Una computadora no tiene sentido común. No tratará de interpretar instrucciones en diferentes formas. No realizará supuestos o llenará los espacios obvios de un algoritmo incompleto.
- Existen eventualidades diferentes a la lógica humana. Por ejemplo: si un algoritmo debe decidir si un estudiante aprueba o reaprueba un curso y se escribe de esta forma:

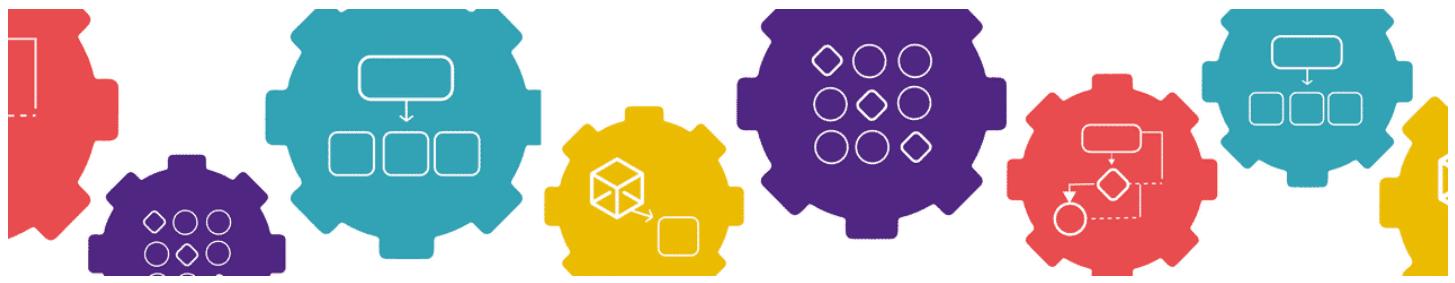
Si (if) la nota es mayor a 61 entonces, marque al estudiante como aprobado.

¿Observa el problema? ¿Qué sucede si el estudiante obtiene como nota 60? Sería obvio para un humano, pero a la computadora se le debe dar la instrucción. El algoritmo quedaría de esta forma:

Si la nota es mayor a 61 entonces, marque al estudiante como aprobado,

Sino (else) marque al estudiante como reprobado.

En el caso de condicionales complejos, se recomienda el uso de paréntesis para explicitar el orden.



## Recursos complementarios

---

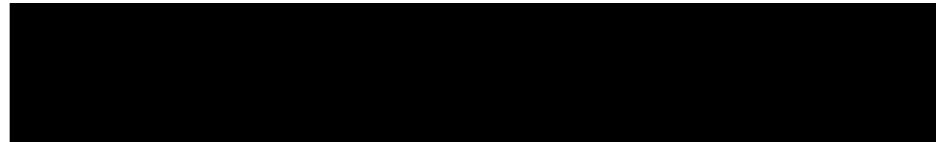


Recursos complementarios

Tu cerebro puede hacer algoritmos

YOUTUBE





## What's an algorithm? - David J. Malan

View full lesson: <http://ed.ted.com/lessons/your-brain-can-solve-algorithms-david-j-malan>  
An algorithm is a mathematical method of solving problems both big a...

[VER EN YOUTUBE >](#)

## Algoritmos

YOUTUBE



## Conceptos Pensamiento Computacional - Algoritmos

Conceptos Pensamiento Computacional - Algoritmos

[VER EN YOUTUBE >](#)



## Actividad 1

---



### Actividad 1



### Instrucciones

Indique si los enunciados son falsos o verdaderos.

Diríjase a la plataforma para realizar la actividad en el espacio correspondiente.



## Actividad 2

---

### Actividad 2



#### Instrucciones

Considere la siguiente receta:

Nombre	Etiquetas	Tiempo de preparación
Ensalada de pollo	Pollo, lechuga, queso gorgonzola, jugo de limón.	15 min
Pavo horneado	Pavo, arroz, cebolla, almendras, ajo.	60 min
Pollo picante	Pollo, jengibre, canela, ajo, frijoles.	30 min
Ensalada de lentejas	Lentejas, cebolla, chiles pimientos, almendras, lechuga.	20 min
Dip de ajo	Ajo, jugo de limón, garbanzos, caldo de pollo.	5 min

Los usuarios pueden buscar recetas según las etiquetas. Considere las siguientes búsquedas y decida qué recetas devolverá el sistema.

- 1 Que se cocine en menos de 20 min y que no sea vegetariano.
- 2 Que incluya pollo y pavo, pero no ajo.
- 3 Que no incluya nueces.

Diríjase a la plataforma para subir su actividad en el espacio correspondiente.

## Actividad 3

---

### Actividad 3



Las siguientes son reglas que aplican para hacer cola en un supermercado:

1. Las personas en la cola que estén sosteniendo sus productos en la mano, toman un minuto para ser atendidos.
2. Las personas que sostienen una canasta toman dos minutos para ser atendidos.

3. Las personas que empujan una carreta toman cinco minutos para ser atendidos si la carreta está medio llena; y toman diez minutos si está llena.

4. Las personas en autoservicio se procesan en 80 por ciento del tiempo de una atención normal.

Exprese estas reglas como enunciados lógicos. Cada uno deberá tener una conclusión del tiempo de atención estimado.

Diríjase a la plataforma para subir su actividad en el espacio correspondiente.

## Actividad 4

---

### Actividad 4



### Instrucciones

Construya un algoritmo a partir de los enunciados de la actividad 3. El algoritmo debe tomar una cola como entrada y la salida debe ser el tiempo estimado total.

Diríjase a la plataforma para realizar la actividad en el espacio correspondiente.

## Actividad 5

---



### Instrucciones

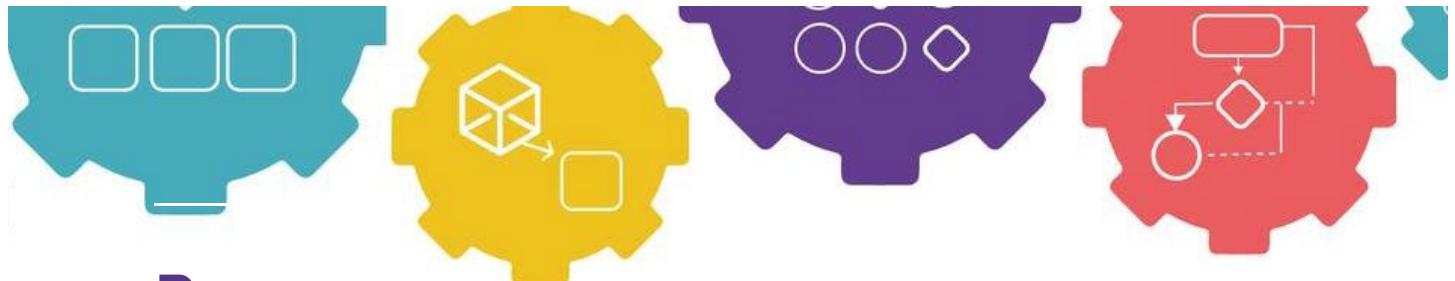
Basado en el algoritmo de 99 Bottles of Beer, escriba un algoritmo para la canción:

- [The 12 Days of Christmas](#)

Diríjase a la plataforma para subir la actividad en el espacio correspondiente.

## Recursos

---

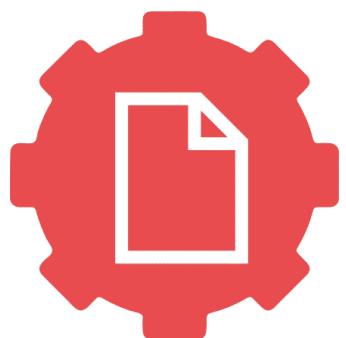


### Recursos

#### Instrucciones:

Tenga a la mano los siguientes recursos a utilizar.





Papel



Computadora



Pizarrón



Lápiz



Acceso a internet



Marcadores

## Rúbrica de evaluación

---

### Rúbrica



Descargue la siguiente rúbrica de evaluación.





Rúbrica de Evaluación.pdf

1423 KB



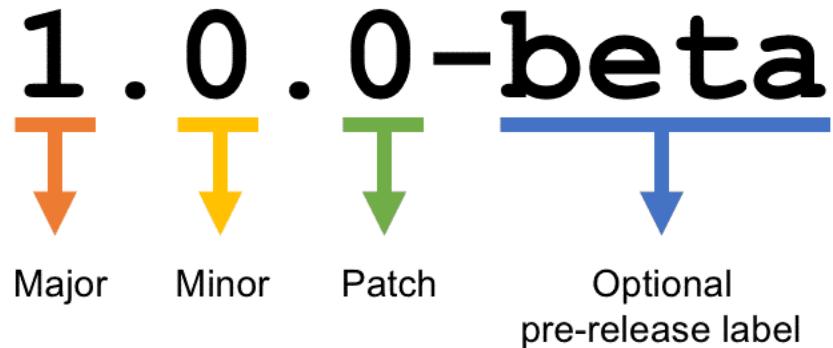
## Diario de experiencias de laboratorio

---



# Diario de experiencias del laboratorio

Cada práctica de laboratorio deberá publicarse en una carpeta en el diario de experiencias de laboratorio, bajo la versión 1.0.0.



## Referencias

---

### Referencias

Bordigon, F. e Iglesias, A. (2020). Introducción al pensamiento computacional. Universidad Pedagógica Nacional, Educar Sociedad del Estado y el Ministerio de Educación Argentina.

Beecher, K. (2017). Computational Thinking. A beginner's guide to problem-solving and programming.

ISTE (2018). Computational Thinking. Meets Students Learning. International Society for Technology in Education.

MIT (2022). Computational Thinking, a live online Julia/Pluto textbook. Julia: A Fresh Approach to Computing. computationalthinking.mit.edu. Massachusetts Institute of Technology.

Pourbahrami & Tritty (2018). Computational Thinking: How Computer Science Is Revolutionizing Science and Engineering. ENGenious (15) 8-11. <https://resolver.caltech.edu/CaltechCampusPubs:20181025-110029152>

## Créditos

---

