compAs
Grupo de capacitación e investigación pedagogica

# Pensamiento comunicacional

Jaime Gabriel Espinosa Izquierdo
John Fernando Granados Romero
Francisco Lenin Morán Peña
Juan Ernesto Fernández Escobar

# Pensamiento comunicacional

Jaime Gabriel Espinosa Izquierdo
John Fernando Granados Romero
Francisco Lenin Morán Peña
Juan Ernesto Fernández Escobar

# Pensamiento comunicacional

compAs
Grupo de capacitación e investigación pedagógica

Título original:
Pensamiento computacional

Primera edición: septiembre 2020
© Jaime Gabriel Espinosa Izquierdo
John Fernando Granados Romero
Francisco Lenin Morán Peña
Juan Ernesto Fernández Escobar
Universidad Estatal de Guayaquil
© 2020,

Publicado por acuerdo con los autores.
© 2020, Editorial Grupo Compás
Guayaquil-Ecuador

Grupo Compás apoya la protección del copyright, cada uno de sus textos han sido sometido a un proceso de evaluación por pares externos con base en la normativa del editorial.

El copyright estimula la creatividad, defiende la diversidad en el ámbito de las ideas y el conocimiento, promueve la libre expresión y favorece una cultura viva. Quedan rigurosamente prohibidas, bajo las sanciones en las leyes, la producción o almacenamiento total o parcial de la presente publicación, incluyendo el diseño de la portada, así como la transmisión de la misma por cualquiera de sus medios, tanto si es electrónico, como químico, mecánico, óptico, de grabación o bien de fotocopia, sin la autorización de los titulares del copyright.

Editado en Guayaquil - Ecuador

ISBN: 978-9942-33-295-0

compÁs
Grupo de capacitación e investigación pedagógica

Cita.
Espinosa. J, Granados. J, Morán. F, Fernandez. J (2020) Pensamiento computacional. Editorial Grupo Compás, Guayaquil Ecuador, 167 pag

# UNIDAD 1


# PENSAMIENTO COMPUTACIONAL

## El pensamiento

El pensamiento es un proceso complejo, de hecho, existen distintos tipos de pensamientos hay un tipo en particular que sin que nos demos cuenta lo utilizamos cada vez que debemos resolver un problema, seguramente lo escuchamos nombrar alguna vez, es el **Pensamiento Computacional**.

## Pensamiento Computacional

El **pensamiento computacional** puede ser desarrollado y aplicado en distintas disciplinas o actividades de la vida cotidiana, lo cual nos plantea un nuevo desafío educativo para nuestros hijos y nuestra sociedad. Por ello es cada vez más necesario introducir el **pensamiento computacional** en el sistema educativo con el objetivo de preparar a los estudiantes para un mercado laboral cada vez más tecnológico, mejorando las habilidades intelectuales y haciendo uso de abstracciones para resolver problemas complejos.

Se denomina pensamiento computacional a un tipo de **pensamiento analítico**. La promotora del pensamiento computacional es Jeanette Wing, directora de Avanessians del Instituto de Ciencias de Datos de la Universidad de Columbia (Nueva York), donde también es profesora de informática.

## Historia

En el año 2006, **Jeannette Wing**, publicó un artículo denominado Computational Thinking para Communications of the ACM, la revista mensual de la Association for Computing Machinery. En el mencionado artículo, Wing expresaba que el pensamiento computacional implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, haciendo uso de los conceptos fundamentales de la informática. Por lo tanto, pensar computacionalmente es pensar como lo haría un científico informático cuando nos enfrentamos a un problema.

Asimismo, planteaba que el **pensamiento computacional** debería ser incluido como una nueva competencia en la formación educativa porque, al igual que la matemática u otra disciplina del saber, es una habilidad fundamental.

El **pensamiento computacional** implica resolver **problemas**, **diseñar sistemas** y **comprender** el comportamiento humano, haciendo uso de los conceptos fundamentales de la informática.

**Es decir, que la esencia del pensamiento computacional es pensar como lo haría un científico informático cuando nos enfrentamos a un problema.**

Otras definiciones del pensamiento computacional han ido surgiendo en la literatura científica desde entonces. Entre las más aceptadas se encuentran la de Aho y la de la Royal Society:

El **pensamiento computacional** es el proceso que permite formular problemas de forma que sus soluciones pueden ser representadas como secuencias de instrucciones y algoritmos.

El **pensamiento computacional** es el proceso de reconocimiento de aspectos de la informática en el mundo que nos rodea, y aplicar herramientas y técnicas de la informática para comprender y razonar sobre los sistemas y procesos tanto naturales como artificiales.

Una iniciativa muy interesante en relación a la definición del **pensamiento computacional** es la promovida por la Sociedad Internacional de la Tecnología en la Educación (**ISTE**) y la Asociación de Profesores de Informática (**CSTA**), que han colaborado con líderes del mundo de la investigación y la educación superior, la industria y la educación primaria y secundaria para desarrollar una definición operativa que describa con precisión sus características esenciales y ofrezca un marco de trabajo y un vocabulario común con el que los profesionales de la educación puedan trabajar.

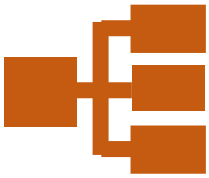
Según esta definición operativa, el **pensamiento computacional** es un proceso de resolución de problemas que incluye las siguientes características:

El objetivo fundamental de **Programamos** es, precisamente, promover el desarrollo del **pensamiento computacional** desde edades tempranas a través de la programación de videojuegos y aplicaciones para móviles en todas las etapas escolares, desde educación infantil hasta formación profesional.

Sin embargo, es posible desarrollar el pensamiento computacional de nuestros estudiantes desde cualquier disciplina y haciendo uso de otros recursos educativos, no solo a través de la programación.

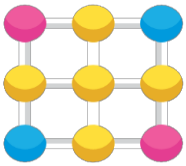
## Elementos del pensamiento computacional

La forma en la cual se resuelven problemas por medio del **pensamiento computacional** consiste en una serie ejes principales para hallar una o varias respuestas a un determinado planteamiento:



**Descomposición del problema:**
Se trata de descomponer grandes problemas en partes más pequeñas que facilitan su resolución.

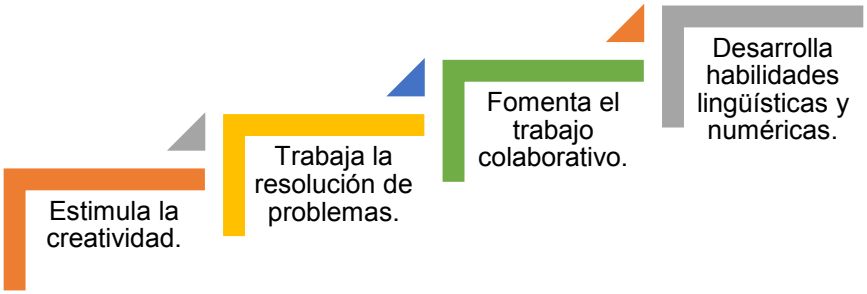
**Reconocimiento de patrones.**
Son patrones similares que se resolvieron anteriormente y se deben integrarlos como parte de la solución del problema.



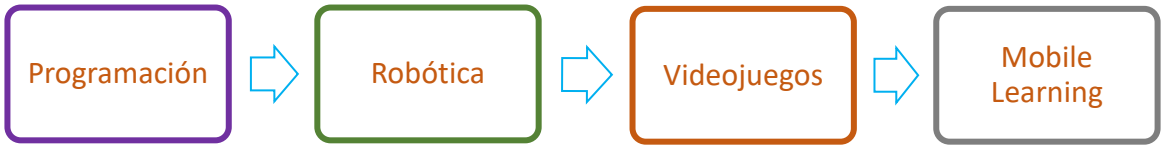
**Realización de abstracciones.**
Reconocer la información necesaria y más relevante descartando detalles innecesarios y abstraer elementos comunes aplicables a otros problemas.

**Diseño de algoritmos.**
Es la definición de los pasos ordenados y lógicos, a seguir para la resolución de un problema.

## Beneficios de usar el pensamiento computacional en clases:

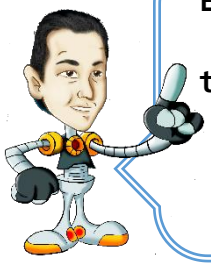



## Algunos recursos para trabajar en el aula de clases:



Países que incluyen el Pensamiento Computacional a nivel Curricular:

Polonia | Dinamarca | Rumania | Australia | Canadá | España | Finlandia | EEUU



**El pensamiento computacional es una metodología que nos permite adaptarnos a los cambios tecnológicos, saber reaccionar a nuevos métodos y por supuesto innovar para encontrar soluciones a los desafíos que nos podemos encontrar a lo largo de la vida.**

## Beneficios del Pensamiento computacional

El pensamiento computacional no es solamente aplicable al mundo informático, al contrario. Los beneficios educativos de poder pensar de manera computacional son varios, empezando por el uso de abstracciones que mejoran y refuerzan las habilidades intelectuales, y que por tanto pueden ser transferidos a cualquier otro ámbito.

Los informáticos ya conocen el valor del pensamiento abstracto, pensando en múltiples niveles de abstracción y empleándola para manejar la complejidad. Nuestra labor incluye difundir los beneficios que el pensamiento computacional puede aportarnos.

El pensamiento computacional implica un conjunto de técnicas y habilidades de resolución de problemas que los programadores utilizan para escribir los programas que conforman las aplicaciones informáticas que utilizamos a diario (buscadores, email, etc). El desarrollo y el trabajo continuado de estas técnicas y habilidades nos proporcionará muchos beneficios tales como:

- Entender qué aspectos de un problema son susceptibles de ser resueltos mediante la computación.
- Evaluar las herramientas y técnicas computacionales más adecuadas a un problema.
- Entender las limitaciones y las capacidades de las herramientas y técnicas computacionales.
- Aplicar o adaptar una herramienta o técnica computacional para un nuevo uso.
- Reconocer las oportunidades de utilizar las técnicas computacionales de una manera novedosa.

- Aplicar las estrategias computacionales como, por ejemplo: divide y conquistaras en cualquier ámbito.

## Importancia del pensamiento computacional

Para definir la importancia del desarrollo del pensamiento computacional, es clave reflexionar sobre el aprovechamiento de la infraestructura tecnológica (los ordenadores) dentro de las aulas como primer punto y como componente que cambia nuestras relaciones sociales, culturales y económicas como segundo punto.

Así, debemos entender que el pensamiento computacional es fundamental para el manejo de la información, la solución de problemas y la comprensión del comportamiento humano. Esto debe desarrollarse preferentemente desde edades tempranas, cuyo desarrollo mejorará la competitividad e innovación, y a la vez formar actitudes y valores en los niños.

Frente a estas ventajas, en los últimos años los sistemas educativos de algunos países han puesto gran énfasis en incluir al pensamiento computacional desde la formación primaria. Este tipo de pensamiento permite aprovechar las ventajas de las transformaciones revolucionarias que los cambios tecnológicos acelerados han producido y hacer además sus propias contribuciones para la solución de los grandes desafíos del Siglo XXI.

## Pensamiento computacional en el sistema educativo

La enseñanza del pensamiento computacional refuerza y estructura de mejor forma los conocimientos que se adquieren en la escuela, además de otorgar muchos otros beneficios a niños y jóvenes estudiantes como:

- Impulsa la confianza en su aprendizaje.
- Desarrolla habilidades blandas o socioemocionales.
- Mejora el entendimiento de materias tradicionales.
- Fomenta la práctica de habilidades STEM.
- Promueve la creación y la innovación.

## Otras definiciones del pensamiento computacional

El **pensamiento computacional** es un proceso cognitivo o pensamiento que implica el razonamiento lógico por el cual los problemas se resuelven y los procedimientos y sistemas se entienden mejor.

La capacidad de pensar de forma algorítmica.

La capacidad de pensar en términos de evaluación.

La capacidad de pensar en términos de descomposición.

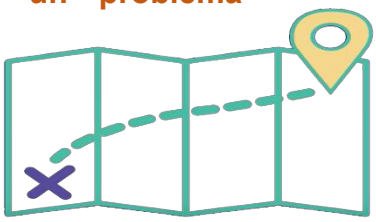
La capacidad de pensar en generalizaciones, identificando y haciendo uso de patrones.

La capacidad de pensar en términos abstractos, la elección de buenas representaciones.

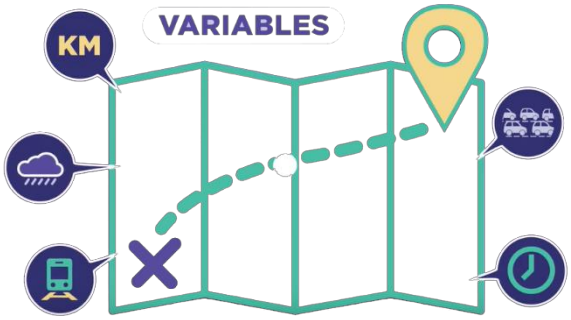
El **pensamiento computacional** también permite, ampliar habilidades de comunicación, aprender a trabajar cono otros para alcanzar un objetivo común lidiar con problemas complejas y situaciones diversas, poner en juego la creatividad, desarrollar el pensamiento crítico y el razonamiento lógico.

**Aplicación del pensamiento computacional a un problema concreto.**

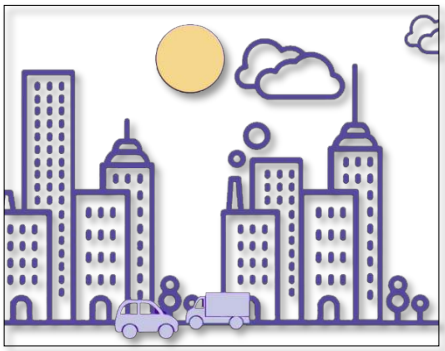
Lo primero que se debe realizar es el **entendimiento** del problema a resolver, ejemplo: como recorrer de un determinado punto de la ciudad a otro lugar.



Se debe tener en consideración las **variables** más significativas como: la distancia, el tiempo, el tránsito, el transporte y el clima.



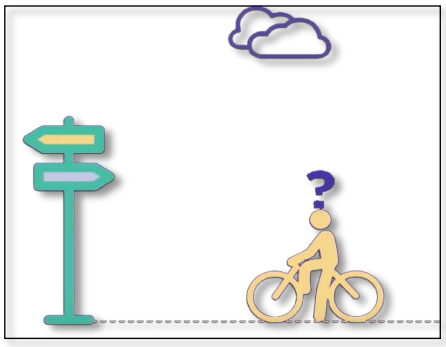

Teniendo estos datos es momento de tomar decisiones:



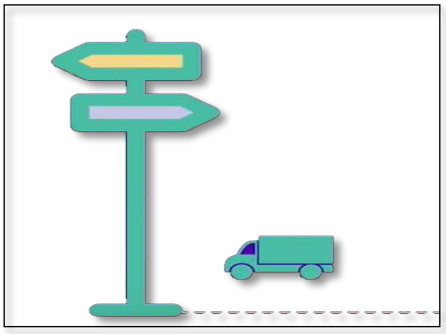
**ES MEJOR TRASLADARME EN VEHÍCULO**

**CAMINANDO O EN BICICLETA**

**EN QUÉ MOMENTO DEL DÍA ES MEJOR**

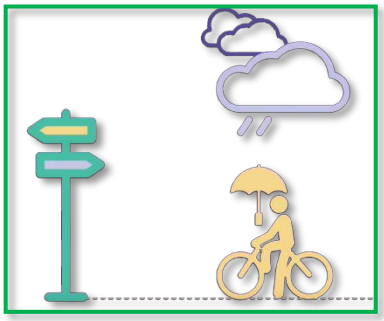


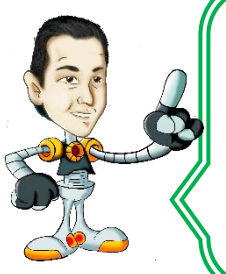
**Y POR CUAL RUTA ES MEJOR.**



Es importante saber nunca existirá una sola solución:

Qué sucede si cambian las variables
Tendremos que cambiar el algoritmo.





**El pensamiento computacional sirve para muchas cosas desde planificar un viaje a organizar un evento.**

## Características del pensamiento computacional

- Organizar y analizar lógicamente la información.
- Representar la información a través de abstracciones como los modelos y las simulaciones.
- Formular problemas de forma que se permita el uso de un ordenador y otras herramientas para ayudar a resolverlos.
- Generalizar y transferir este proceso de resolución de problemas para ser capaz de resolver una gran variedad de familias de problemas.
- Identificar, analizar e implementar posibles soluciones con el objetivo de lograr la combinación más efectiva y eficiente de pasos y recursos.
- Automatizar soluciones haciendo uso del pensamiento algorítmico (estableciendo una serie de pasos ordenados para llegar a la solución).

## Donde utilizamos el pensamiento computacional

El objetivo fundamental de Programamos es, precisamente, promover el desarrollo del **pensamiento computacional** desde edades tempranas a través de la programación en todas las etapas escolares, desde educación infantil hasta formación profesional. Sin embargo, es posible desarrollar el **pensamiento computacional** de nuestros estudiantes desde cualquier disciplina y haciendo uso de otros recursos educativos, no solo a través de la programación.

## Pensamiento algorítmico

El **pensamiento algorítmico** es la capacidad de pensar en términos de secuencias lógicas y ordenadas, siguiendo una serie de reglas para resolver algún tipo de problema o situaciones de entendimiento. Es un conocimiento esencial que los estudiantes adquieren y desarrollan cuando aprenden a escribir sus propios programas en un determinado lenguaje de programación.

**Descomposición:** La descomposición es una forma de pensar acerca de ciertas cosas en términos de sus partes y componentes. Cada pieza debe entenderse, solucionarse, desarrollarse y evaluarse por separado; esto hace más fácil de resolver problemas.

**Por ejemplo:**

Haciendo el desayuno se puede dividir, o descomponer, en actividades separadas tales como:

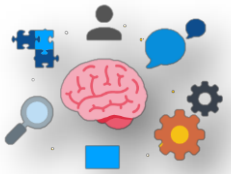
Hacer tostadas

Haz Te o café

Freír el huevo

Cada uno de estas acciones, a su vez, podría también ser dividido en una serie de pasos. A través de la **descomposición** de la tarea original cada parte puede ser desarrollada e integrada más tarde en el proceso; considerar el desarrollo como un juego: una persona puede diseñar y crear los diferentes niveles de forma independiente, siempre que los aspectos clave se acuerdan de antemano.

Es una forma de resolver rápidamente los nuevos problemas sobre la base de las soluciones en los problemas anteriores, y la construcción en la experiencia previa. Haciendo preguntas tales como: ¿Esto es similar a un problema que ya he solucionado?



Y ¿Cómo es diferente? Son importantes aquí, como es el proceso de reconocimiento de patrones, tanto en los datos utilizados y se utilizan los procesos – estrategias.

Algoritmos que resuelven algunos problemas específicos se pueden adaptar para resolver toda una clase de problemas similares. Entonces cada vez que se presenta un problema de esa clase, la solución el general puede ser aplicada.

**Por ejemplo:**
Un estudiante utiliza un juego geométrico para dibujar una serie de formas, tales como un cuadrado y un triángulo, luego escriben un programa para dibujar las dos formas. Ahora quieren dibujar un octágono y una forma de 10 caras. A partir del trabajo con el cuadrado y el triángulo, detectaron que hay una relación entre el número de las dos formas y los ángulos involucrados.

## Técnicas asociadas con el pensamiento computacional

Hay una serie de técnicas empleadas para demostrar y evaluar el **pensamiento computacional**. Pensar en esto como tarea computacional. Estos son el equivalente del **método científico** en **Ciencias computacionales**. Estas son las herramientas con las que el **pensamiento computacional** se opera en el aula, lugar de trabajo y el hogar.

Un elemento esencial del desarrollo de cualquier sistema informático traduce el diseño en forma de código y evaluarlas de manera a garantizar que funcione correctamente en todas las condiciones. La depuración es la aplicación sistemática de las habilidades de análisis y evaluación utilizando como prueba, la localización, y el pensamiento lógico para predecir y verificar los resultados.

Pseudocódigo, diagramas de sistemas, son nuevas actividades de descomposición, y la abstracción en el diseño de algoritmos. Analizar consiste en dividir en partes todos los componentes (descomposición), reduciendo la innecesaria complejidad (abstracción), la identificación de los procesos (algoritmos) y la búsqueda de elementos comunes o patrones (generalización). Se trata de utilizar el pensamiento lógico tanto para comprender mejor las cosas como de evaluar su adecuado propósito.

## El pensamiento computacional en el aula

En el mundo actual, vivimos continuamente rodeados de las últimas tecnologías. Solo hace falta ver el gran uso que hacemos de ellas en nuestra vida diaria. La costumbre de utilizar la tecnología hace que desarrollemos, consciente e inconscientemente, nuevas formas de entender la realidad, nuevas habilidades y competencias y nuevas formas de pensar. De esta forma, nace lo que se conoce como **pensamiento computacional**.

Entonces, se identifican distintos comportamientos que se pueden observar en el aula, como el pensamiento algorítmico que es la capacidad de pensar en términos de secuencias y las reglas son como una forma de resolver problemas.
Es un conocimiento esencial que los estudiantes desarrollan cuando aprenden a escribir sus propios programas en un determinado lenguaje de programación en el ordenador.

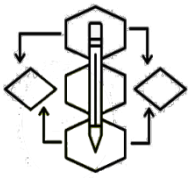
Pasos para cómo resolver un problema en el aula:

- Conformar equipos de trabajo.
- El primer grupo formula las instrucciones, para lograr el efecto deseado.
- Determinar la secuencia ordenada y lógica de pasos a seguir en la formulación de instrucciones.
- Fijar las instrucciones para el desarrollo de operaciones aritméticas, matemáticas y lógicas.
- Escribir secuencias de instrucciones que almacenan, mueven y manipulan los datos (variables y asignación).

# UNIDAD 2

# ALGORITMOS

## Algoritmo



El **algoritmo** es una secuencia ordenada y lógica de pasos a seguir para resolver un problema y deben ser enumerados, pueden ser de distintos tipos, los de la vida diaria también conocidos como **domésticos,** ejemplo: como seguir las instrucciones para lavarse los dientes, algoritmos **lógicos**, utilizan la lógica para resolverlos problemas más complicados y finalmente algoritmos **aritméticos**, son utilizados para resolver problemas que se relacionan con procesos matemáticos.

## Características de los algoritmos

1. Debe definir tres partes: Entrada, Proceso y Salida
2. Debe ser preciso: tiene enumerarse cada paso a realizar.
3. Debe estar definido: si se sigue un algoritmo dos veces, se debe obtener el mismo resultado
4. Debe ser finito: el algoritmo se debe terminar en algún momento
5. Debe ser legible: el texto que lo describe debe ser claro, tal que permita entenderlo y leerlo fácilmente

## Tipos de algoritmos

**Algoritmos Domésticos.** Son aquellos que se utilizan para resolver los problemas de la **vida cotidiana.** Usualmente no nos sentamos a escribir un algoritmo antes de resolver un problema doméstico. Ejemplo: Imagine que usted va viajando en su vehículo tranquilamente vía a Salinas. De pronto, una llanta parece ponchada. Se detiene, examina la llanta y comprueba su sospecha. La llanta está tubo abajo. A continuación, un algoritmo para cambiar la llanta y seguir el viaje hacia las hermosas playas ecuatorianas.

1. Sacar la llave de cruz.
2. Aflojar las tuercas de la llanta.
3. Sacar la gata.
4. Colocar la gata debajo del auto.
5. Accionar la gata para que el auto se eleve.
6. Sacar las tuercas.
7. Sacar la llanta ponchada.
8. Bajar la llanta de emergencia.
9. Colocar la llanta de emergencia.
10. Colocar las tuercas.
11. Bajar la gata.
12. Ajustar las tuercas de la llanta.
13. Guardar las herramientas (gata y llave).
14. Guardar la llanta ponchada.
15. Continuar el viaje.
16. Fin.

Este es el algoritmo correcto para cambiar la llanta.

Nótese que pudieron ser más de 16 pasos si hubiéramos sido mucho más detallistas, como, por ejemplo, ajustar la **1ª. Tuerca, ajustar la 2ª. Tuerca, etc.**

Sin embargo, el número de pasos depende del criterio del programador. Recuerde, un buen algoritmo debe ser sobre todas las cosas **lógico y ordenado.**

El siguiente algoritmo permite servir una taza de café para el desayuno:

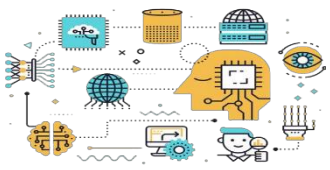
1. Hervir suficiente agua.
2. Llenar una taza de agua hervida.
3. Colocar una cucharadita de café.
4. Colocar dos cucharadas de azúcar.
5. Revolver bien.
6. Tomar café.
7. Fin.

Este algoritmo puede variar si a Ud. le gusta el café más cargado o con menos azúcar, etc.

Para el siguiente algoritmo solicitamos ayuda a una señorita. Mientras la hermana mayor se maquilla para ir a una elegante fiesta, su hermana menor la observaba para luego tratar de imitarla. Sin embargo, la joven notó una gran cantidad de pasos. Entonces decidió escribirlos para no olvidarlos:

1. Quitar residuos de maquillaje anterior (con crema).
2. Lavarse la cara.
3. Secarse la cara.
4. Aplicar la base o polvo compacto.
5. Aplicar el blush.
6. Delinear las cejas.
7. Aplicar la sombra.
8. Delinear los párpados.
9. Rizar las pestañas.
10. Aplicar el rime.
11. Delinear los labios.
12. Fin.

Como podrá apreciar, en cada momento de nuestras vidas utilizamos algoritmos para la resolución de problemas domésticos. A continuación, otro tipo de algoritmos.



**Algoritmos Lógicos.** Son aquellos que para su resolución necesitamos la ayuda de algún artificio lógico y de razonamiento pausado y calculado del problema.

**Ejemplo:**

Suponga que tenemos tres copas **A**, **B**, **C**. En la copa **A** tenemos café; en la **B** tenemos vino; la copa **C** está vacía. Queremos intercambiar los contenidos de las copas A y B, es decir café en B y vino en A.


1. Intercambiar A en C.
2. Intercambiar B en A.
3. Intercambiar C en B.
4. Fin

**¡Muy sencillo…! Veamos otro ejemplo:**

En la cumbre de una montaña se encuentran tres alpinistas. Dos de ellos pesan 60 Kg. cada uno, mientras que el tercero pesa 120 kg. Desean pasar a la cumbre de la otra montaña, y para ello disponen de un transportador manual que solo soporta un peso máximo de 120kg. Escriba un algoritmo que especifique las secuencias de pasos a seguir para que los tres alpinistas pasen a la otra montaña.

1. Pasan los dos delgados.
2. Regresa un delgado.
3. Pasa el gordo.
4. Regresa el otro delgado.
5. Pasan los dos delgados.
6. Fin.

Obviamente el alpinista delgado (60kg) tiene que regresar dos veces debido al que el transportador requiere de alguien que lo conduzca.

Un hombre deseaba transportar un lobo, una gallina, y un saco de maíz de una orilla a otra. Dispone de una canoa que sólo resiste su peso (el del hombre) y de otro más (lobo, gallina o maíz), es decir, sólo dos pesos. Elabore un algoritmo que indique las secuencias de pasos a seguir para que el hombre transporte al lobo, gallina y maíz, sin que en ninguna de las dos orillas queden solos el lobo y la gallina (porque se la come) ni la gallina sola con el maíz (porque también se lo come). Suponiendo que es el hombre el que conduce la canoa:

1. Pasa con la gallina.
2. Pasa el maíz, pero se regresa con la gallina.
3. Deja la gallina y pasa al lobo.
4. Pasa con la gallina.
5. Fin.

Se tiene dos recipientes: uno para capacidad de 5lts. y el otro con capacidad de 4lts. Se desea tener exactamente 3litros en el recipiente de 4lts, sabiendo que disponemos de abundante agua, y los recipientes no tienen marcación alguna.

1. Se llena el recipiente de **4lts**.
2. Se vierte este contenido en el de **5lts**.
3. Se llena nuevamente el recipiente de **4lts**.
4. Se vierte el contenido en el recipiente de **4lts**, en el de **5lts** hasta que este último se llene, como le faltaba sólo un litro. En el recipiente de 4lts, exactamente quedan **3lts**.
5. Fin.

**Algoritmos matemáticos.** Son utilizados en la solución de problemas aritméticos y que tiene que ver con la aplicación de una fórmula matemática. Un algoritmo matemático requiere saber cuáles son los datos con que contamos, cuál va a ser la fórmula empleada y el resultado que deseamos obtener, el siguiente algoritmo matemático define los pasos para obtener el promedio trimestral de un estudiante

1. Obtener el 1er aporte.
2. Obtener el 2do aporte.
3. Obtener el 3er aporte.
4. Saber la calificación del examen.
5. Sumar todas las notas.
6. Promediar para 4.
5. Fin.

Se desea calcular la hipotenusa de un triángulo rectángulo. Recuerde que la fórmula es:

$$H = \sqrt{a^2+b^2}$$

1. Saber cuánto vale el cateto mayor.
2. ¿Cuánto vale el cateto menor?
3. Elevar el cuadrado del cateto mayor.
4. Elevar el cuadrado del cateto menor.
5. Sumar estos dos resultados.
6. Extraerle la raíz cuadrada.
7. Fin.

El siguiente algoritmo que establece el salario a percibir por un empleado que trabaja por horas:

1. Determinar cuántas Horas ha trabajado el empleado.
2. Saber cuánto gana por hora.
3. Multiplicar las horas trabajadas por el sueldo de horas.
5. Fin.

## PSEUDOCÓDIGOS

El Pseudocódigo es una **versión abreviada,** de fácil entendimiento y en un **Lenguaje común y corriente** del programa final. En un Pseudocódigo se desglosa el programa sin interesar el lenguaje de programación que se vaya a utilizar. Un Pseudocódigo consta de **palabras en español** describiendo las acciones que se deben efectuar para resolver el problema. Se utilizan palabras tales como: **lea, ingrese, haga, calcule, repetir, si, mientras**, entre otras.

**Realice un pseudocódigo que ingrese la edad de una persona.**

**Inicio**
    Escribir: "¿Cuál es tu edad?
**Leer** Edad
    Escribir "Tu edad es" + Edad
**Fin**

Primero nos pide ingresar la edad, luego escribiríamos la edad y el valor introducido lo recogería la variable **Edad** (imagina que introducimos **18**). Por último, mostraría en pantalla la frase: **Tu edad es 18.**

**A continuación, realizamos un ejercicio que ingresa dos números y obtiene la suma.**

**Inicio**
    Escribir ("Ingrese dos números para sumar: ");
**Leer** (Numero1, Numero2);
    SU = Numero1 + Numero2;
    Escribir ("La suma es: ", SU);
**FIN**

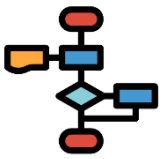
Para este ejercicio ingresamos dos números desconocidos, que se almacenan en las variables **(Numero1, Numero2);** luego realizamos una operación matemática para este caso es una suma **SU = Numero1 + Numero2**; y finalmente presentamos el resultado de la suma **("La suma es: ", SU);**

# UNIDAD 3

# DIAGRAMAS DE FLUJO

## Diagrama de flujo o Flujograma

Un diagrama de flujo es la resolución ***gráfica*** de un problema de programación. El diagrama de flujo o Flujograma consta de una serie de **símbolos** especiales los cuales representan cada uno de los pasos o acciones a seguir para resolver el programa.



Entre las principales ***ventajas*** de los diagramas de flujo tenemos:

Permite tener una ***visión*** mucho más ***amplia*** y ***objetiva*** del programa a codificar

Un mismo diagrama de flujo puede ser luego codificado indistintamente a ***cualquier lenguaje*** **de programación de alto nivel.**

Es mucho más ***fácil y rápido codificar*** un programa teniendo como base el diagrama de flujo**.**

Los diagramas de flujo sirven como una excelente documentación a la hora de hacer **modificaciones** en el programa. Es más práctico efectuar primero las modificaciones en el diagrama que en la codificación**.**

La descripción paso a paso de todas las instrucciones del programa resulta de máxima utilidad para el programador, ya que es el fiel reflejo del programa. Es de gran ayuda para ***estructurar*** su propia documentación de referencia**.**

Dicho de otra manera, el **diagrama de flujo** es tan importante para el programador como un ***plano*** de un edificio para un constructor. Ud. cree que un Ingeniero Civil podría edificar una casa sin tener previamente un plano de la construcción?

Bueno, un programador no puede (ni debe) escribir la codificación de un programa sin haber diseñado antes el ***diagrama de flujo.***

## Simbología

A continuación, los **principales** símbolos utilizados en los diagramas de flujo:

**Inicio/Fin**

Indica el ***inicio*** o ***fin*** de un diagrama. Todo diagrama por extenso o corto que sea, debe ***empezar*** y ***terminar*** con este símbolo.

**Entrada Manual**

Especifica que se van a ***ingresar datos*** por medio del teclado. Se utiliza con palabras tales como lea, ingrese, introduzca.

**Salida por Pantalla**

Se utiliza para **visualizar**, **presentar** o **mostrar** todo tipo de datos por pantalla.

**Proceso** o Calculo

Se utiliza este símbolo cuando se emplea alguna **fórmula** para resolver un proceso o cálculo matemático

**Estructuras Selectivas**

Una Estructura Selectiva es una **condición** o **pregunta** que puede tener solo dos opciones de respuestas: verdadero o falso.

**Ciclo**

El símbolo de **ciclo** o **bucle** se usa cuando se desea **repetir** un determinado número de veces una parte del diagrama de flujo

**Salida por impresora**

Símbolo usado para representar la **salida** o presentación de datos por **impresora.**

**Conector Interno**

Cuando un diagrama es demasiado largo se utiliza el conector interno para especificar que el diagrama **continúa** en otra parte de la **misma hoja**.

**Conector Externo**

Cuando un diagrama requiere continuar en **otra hoja**, se utiliza el conector externo.




Este es un símbolo que se puede usar indistintamente para representar un **ingreso** o una **salida de datos**.

**Líneas de Flujo**

Permite **unir** o enlazar los símbolos del diagrama entre sí


## Variables

Una variable es un ***casillero de memoria*** en el cual podemos almacenar datos. Estos datos pueden variar dependiendo de la ejecución del programa. Toda variable debe tener un nombre. Los nombres de las variables deben cumplir con las siguientes reglas:

Deben de empezar siempre con una ***letra***, (Mayúscula o Minúscula) y pueden ser una ***combinación*** de ***letras*** y ***números***. Por ejemplo, nombres correctos de variables son: SUMA, XX, M1, M2, XYZ123, SUELDO, entre otros.

**No** pueden tener ***espacios en blanco,*** ni símbolos especiales intermedios. Para nombres de variables largos, use abreviaturas (sin puntos). Por ejemplo: incorrecto es usar SUELDO POR HORA, correcto sería SUELXHOR.

Deben ser nombres ***cortos*** **y** significativos. Evite usar nombres largos y difíciles. Por ejemplo, en lugar de usar la variable PROMEDIO use PROM.

Cada nombre de variable debe ser ***único***, es decir, no puede existir otra variable con el mismo nom

## Operadores aritméticos

Los operadores aritméticos permiten efectuar ***procesos*** o cálculos ***matemáticos elementales.*** Los operadores aritméticos son:

| OPERADOR | OPERACIÓN | EJEMPLO |
|---|---|---|
| + | Suma | 5 + 2 = 7 |
| - | Resta | 5 – 2 = 3 |
| * | Multiplicación | 5 * 2 = 10 |
| / | División | 5 / 2 = 2.5 |
| Mod | Residuo de división | 5 mod 2 = 1 |

En el caso de los operadores "/" y Mod, son exclusivamente para la división. El operador "/ "es usado para la división decimal, es decir, la división común y corriente. Por ejemplo:
6 / 3 = 2
7 / 2 = 3.5
6 / 4 = 1.5
4 / 3 = 1.33333

El operador Mod obtiene el residuo, es decir, lo que sobra de una división. Por ejemplo:

14 Mod 3 = 2 (porque 14 / 3 es 4 sobrando 2)
6 Mod 4 = 2 (divida 6 para 4 y le sobran 2)
9 Mod 2 = 1 (sobra 1 al dividir 9 para 2)
10 Mod 5 = 0 (divida 10 para 5 y no le sobra nada)

Adicionalmente existe la función INT, la cual permite extraer la parte entera de una expresión decimal, es decir desecha los decimales (si los hubiere). Por ejemplo:

INT (13 / 3) = 4 (13/3 es 4.33, pero INT solo toma en cuenta la parte entera)
INT (7 / 2) = 3
INT (6 / 4) = 1

## Reglas de prioridad

Dentro de una expresión matemática compleja, siempre se efectúan primero los paréntesis más internos, luego las divisiones y las multiplicaciones y al final se realizan las sumas y restas.
Por ejemplo: 5 + 4 / 2 – 1
Primero se efectúa 4 / 2, o sea 2 y luego 5 + 2 - 1. El resultado es 6.

| | |
|---|---|
| **( )** | **Prioridad Máxima** |
| **/, Mod, *** | **Prioridad Intermedia** |
| **+, -** | **Prioridad Mínima** |

Por ejemplo:
5 + 2 * 4 - 3
Es 10; primero se efectúa 2*4 y luego se suma y se resta. Sin embargo, los paréntesis siempre se evalúan primeros.

Por ejemplo:
(5 + 2) * 4 - 3
Es 25; primero se hace el paréntesis (5+2) y luego se multiplica por 4 y se resta 3.

Veamos ahora los siguientes ejemplos:

A) 2 + 1 * 5 + 2
Es 9, porque primero es 1 * 5 = 5; luego 2 + 5 + 2 = 9

B) (5 Mod 2) + 2 / 2
Primero es 5 Mod 2 = 1 y 2 / 2 = 1; entonces 1 + 1 = 2

C) INT (11 / 3) – (7 Mod 2)
INT (11 / 3) es 3 y 7 Mod 2 es 1; entonces 3 – 1 = 2

D) 5 + 2+3 / 2
Primero 3 / 2 = 1.5; más 5 + 2 + 1.5 = 8.5

## Fórmulas

Una fórmula es una expresión que se utiliza para obtener un resultado. Por ejemplo:
SU = PV + SV

Es la fórmula que representa la suma de dos valores, donde la variable "PV" representa el primer valor, la variable "SV" el segundo valor y la "SU" la suma.

Las fórmulas siempre se evalúan de derecha a izquierda, es decir, colocando la expresión a calcular al lado derecho del igual y la variable que guarda el resultado del lado izquierdo.

La siguiente fórmula representa el promedio de tres calificaciones de un estudiante:

Expresión a calcular

Resultado a Obtener PRO = (PN + SN + TN) / 3

Sin embargo, no todas las fórmulas llevan exclusivamente variables. En el caso de conversiones o transformaciones se utilizan valores constantes.

Una constante es un valor preestablecido, es decir, que no cambia durante la ejecución del programa.

Ejemplos de valores constantes son:

- Los Minutos que tiene una hora = 60
- Los segundos de una hora = 3600
- Las horas de un año = 8760
- Los días del año = 365
- Los días de la semana = 7
- Las semanas del año = 52
- Los meses de año = 12
- Metros que tiene un Kilómetro = 1000
- Libras que tiene un kilo = 2.2;

## Flujogramas de procesos simples

Ahora vamos a realizar nuestro **primer diagrama** de flujo. Son diagramas de procesos simples aquellos en los que se ingresan una o más variables, luego se aplica alguna fórmula y finalmente se visualiza o imprime el resultado. Lea detenidamente el enunciado del problema:

Elabore un diagrama de Flujo que lea dos valores. **Calcule** y **Visualice** su suma.

Bien. Analicemos el enunciado. Primero ya sabemos que todo diagrama debe empezar con el símbolo de **Inicio (fig1).** Luego observe como se pide que **lea** dos valores.

Entonces necesitamos dos variables; por ejemplo, **A** y **B** y las colocamos dentro símbolo de ingreso (**fig2**). Después se pide que se calcule su suma. Entonces aplicamos la fórmula **S = A + B** y la ponemos dentro del símbolo de **proceso (fig3).**

Al final **visualiza** su suma. Entonces colocamos la variable **S** que representa la suma, dentro del símbolo de visualizar (**fig4**). Para terminar, colocamos el símbolo de **fin** (**fig5**) y unimos todos los símbolos con líneas de flujo y tenemos listo nuestro diagrama.

**Figura 1**

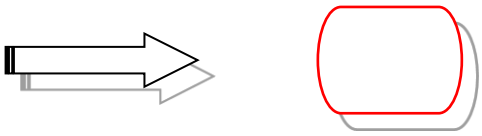


**Figura 2**

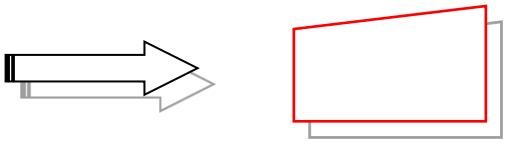


**Figura 3**

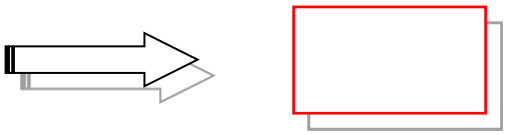


**Figura 4**

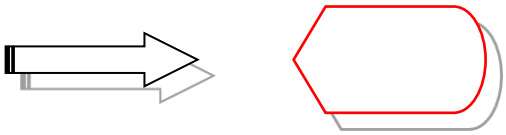
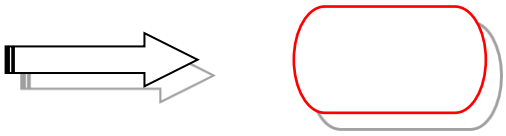
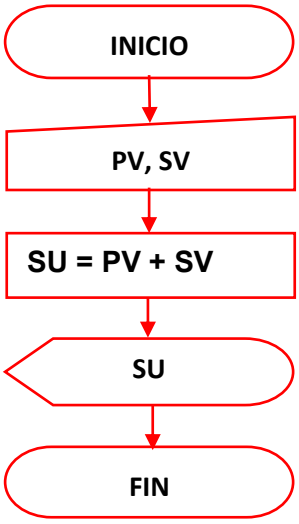


**Figura 5**



## Ejercicio 1

Elabore un diagrama de Flujo que lea dos valores. **Calcule y visualice** su suma.




Las **variables** usadas van a representar los dos valores ingresados. "**PV**" representa al ***primer valor*** y "**SV**" representa al ***segundo valor.***

Luego, mediante la fórmula **SU = PV + SV** calculamos la suma y se almacena en la variable "**SU**". Siempre el resultado de todo cálculo debe almacenarse en alguna variable de memoria,

A continuación, presentamos el resultado de la suma **SU** mediante el símbolo de visualizar.

Para comprobar que el diagrama de flujo esté bien hecho, se le

La **prueba** de **escritorio** consiste en darle valores ficticios o de prueba a todas las variables de entrada, es decir en este caso **PN** y **SN** luego se simulan los cálculos y se obtienen los resultados.

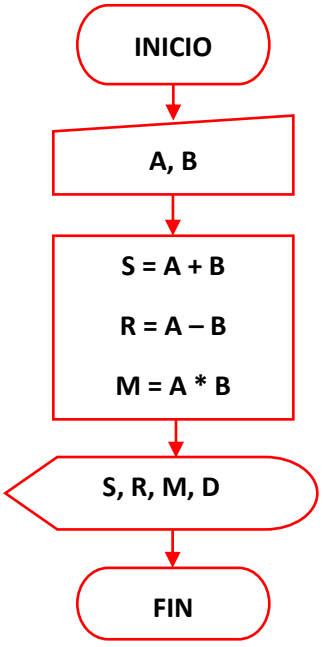
**PRUEBA DE ESCRITORIO**

| PV | SV | SU |
|---|---|---|
| 3 | 5 | 8 |
| 10 | 2 | 10 |

Los valores de 3, 5, 10 y 2 son valores ficticios, usted puede colocar sus propios valores.

## Ejercicio 2

Lea dos números. Visualice los resultados de sus 4 operaciones fundamentales:




Para los dos números ingresados usaremos las variables **A** y **B**.

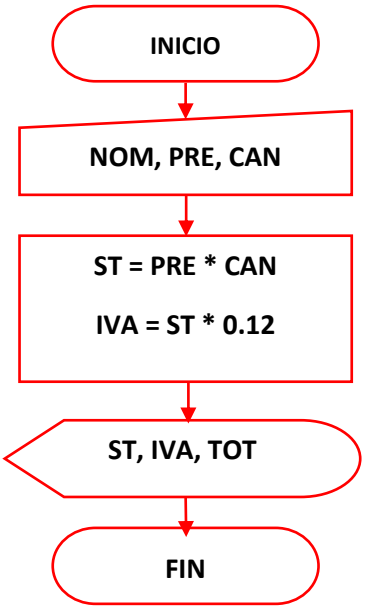
Luego almacenamos el resultado individual de cada operación en una variable distinta; así **S** para la suma, **R** para la resta, **M** para la multiplicación y **D** para la división.

Finalmente presentamos los resultados, es decir, **S, R, M, D.** Veamos la prueba de escritorio:

| A | B | SU | RE | MU | DI |
|---|---|---|---|---|---|
| 6 | 3 | 9 | 3 | 18 | 2 |
| 21 | 7 | 28 | 14 | 147 | 3 |

## Ejercicio 3

Ingrese el nombre, el precio y la cantidad de un producto a comprar. Calcule y visualice el subtotal, el 12% IVA y el total a pagar.




**CÁLCULO DE PORCENTAJE.**

Un porcentaje es la **proporción** de una cantidad con relación a otra que se calcula sobre la centena. De esta manera:

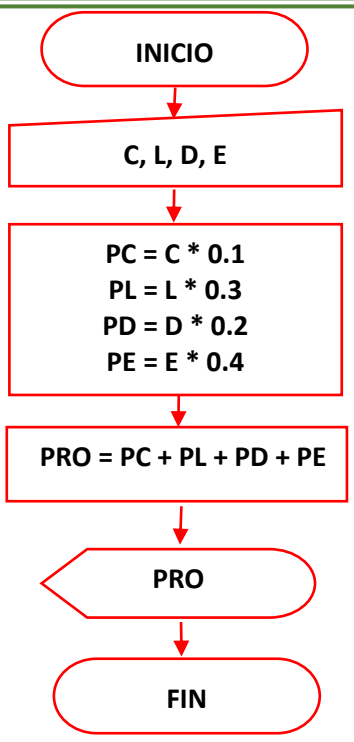
Porcentaje = tanto * monto / 100

Donde **tanto** es el tanto por ciento a aplicar; **monto** la cantidad a la que se le extrae el tanto y el **porcentaje** el resultado obtenido. De esta manera veamos el 5% de 2000:

Porcentaje = 5 * 2000 / 100

Porcentaje = 100

Usaremos las variables **NOM** para el nombre del producto, **PRE** para el precio y **CAN** para la cantidad. El cálculo del subtotal se obtiene multiplicando el precio por la cantidad. El 12% IVA se obtiene multiplicando el subtotal por **0.12;** el total a pagar es la suma del subtotal más el IVA. Luego presentamos todos los resultados obtenidos.




## Ejercicio 4

En cierto colegio, el promedio trimestral se calcula de la siguiente manera: el 10% de la nota es el cuaderno al día, el 30% las lecciones, el 20% los deberes y el 40% el examen. Si todas las calificaciones son sobre 20, elabore un diagrama que ingrese las notas de cuaderno, lección, deberes y exámenes de un estudiante. Visualice el promedio trimestral.

Ingresemos las variables **C** cuaderno, **L** lecciones, **D** deberes, **E** exámenes. Estas notas son sobre 20 y hay que extraerle el porcentaje para obtener el promedio trimestral. Entonces multiplicamos cada nota por su respectivo porcentaje. **PC** porcentaje de cuaderno, **PL** porcentaje de lección, **PD** porcentaje de deberes y **PE** porcentaje de exámenes. Luego sumamos los resultados obtenidos y presentamos el promedio.

## Estructuras de control

Una estructura de control, es una pregunta con tan solo dos respuestas posibles: **Verdadero** o **Falso**. Las Estructura control se construyen a partir de las condiciones.

Una **condición** es una expresión lógica o una pregunta de forma:

Variable ***OPERADOR RELACIONAL*** Variable/valor

## Operadores

Los Operadores Relaciónales son:

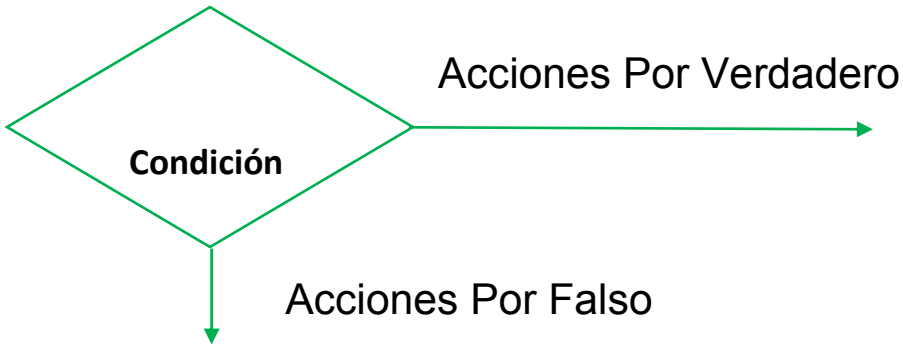
| OPERADORES RELACIONALES | SIGNIFICADO | EJEMPLO |
|---|---|---|
| **>** | Mayor que | 5 > 2 |
| **<** | Menor que | 3 < 6 |
| **=** | Igual | 1 = 1 |
| **>=** | Mayor o igual | 4 >= 3 |
| **<=** | Menor o igual | 5 <= 10 |
| **< >** | Diferente a | 2 < > 7 |

De esta manera, utilizamos los operadores lógicos y podemos construir las siguientes condiciones, suponiendo que: **A = 1;** **B = 2** **y** **C = 0;**

**a)** **A > B**
**b)** **C < A**
**c)** **C = 0**
**d)** **A >= 1**
**e)** **B <= 0**

El **literal A)** es falso, porque **"A "**que vale 1 no es mayor que **"B"** (que vale 2).
El **Literal B)** es verdadero porque **"C"** vale 0 y A vale 1.
El **Literal C)** es verdadero porque "C" = 0.
El **Literal D)** es verdadero porque **"A"** que vale 1 si es mayor o igual que 1.
El **Literal E) es** falso porque **"B"** no es menor o igual que 0.
Por último, el **Literal F)** es verdadero porque **"A"** si es diferente que **"C"**.

En un diagrama de flujo las condiciones se representan mediante el símbolo de la Estructura Selectiva, indicando claramente cuáles serán las dos alternativas de Verdadero o Falso.



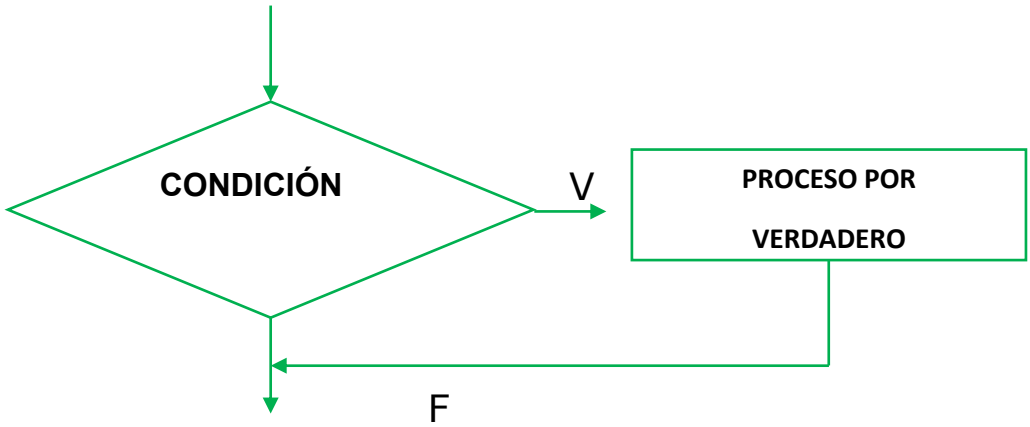

## Tipos de Estructuras de Control

Estructura de Control Simple.
Estructura de Control Dobles.
Estructura de Control Múltiples.
Estructura de Control Compuestas.

### Estructura de Control Simple

Toda Estructura de Control tiene dos alternativas: **verdadero** y **falso**. Sin embargo, las Estructura de Control **simple** sólo realizan acciones cuando la **Condición es Verdadera**, cuando la **Condición es Falsa** no se realiza ninguna acción, es decir, el diagrama continúa su flujo normal.

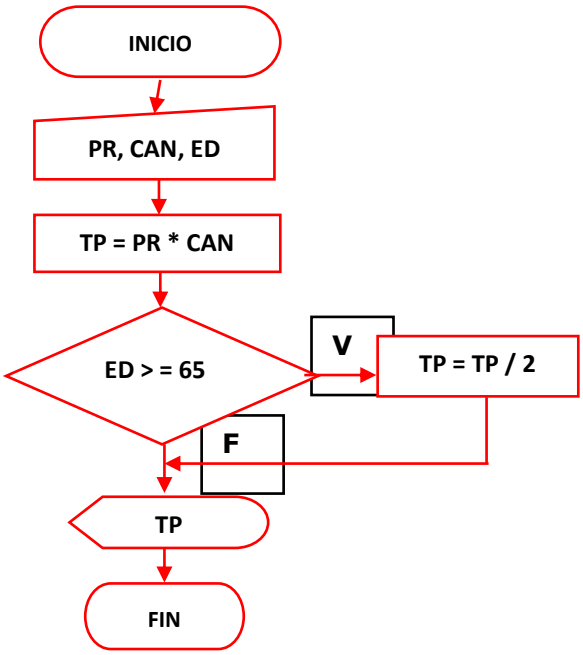
Las **Estructura** de Control **Simple** son de la siguiente forma:




Nota: cuando la condición es ***verdadera***, se realiza un proceso, sin embargo, cuando la condición es ***falsa***, no se realiza nada.

## Ejercicio 1

Elabore un diagrama que lea el precio individual y el número de entradas al cine a comprar. Adicionalmente ingrese la edad de la persona. Calcule y visualice el total a pagar. Considere que si la persona es de la 3ra. Edad (65 años o más) debe pagar sólo la mitad de todo.

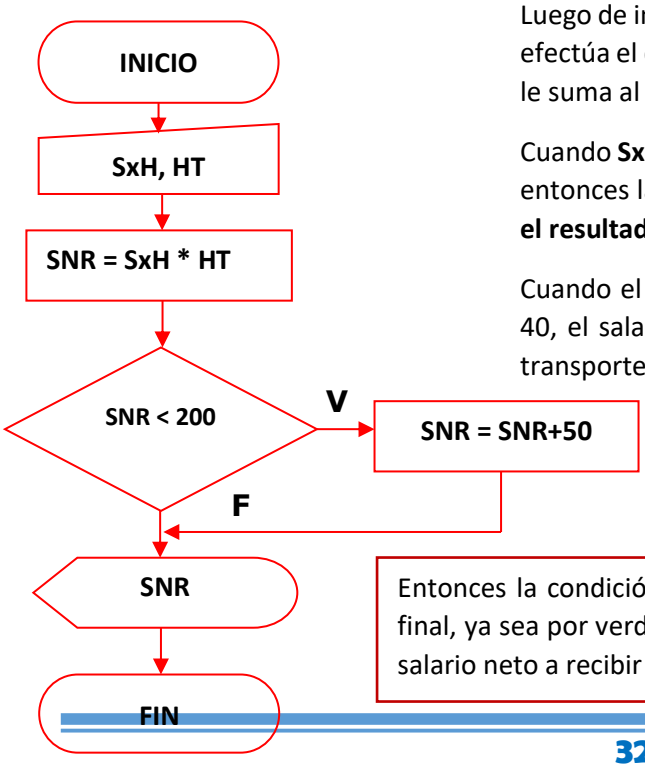



Se ingresa el precio y la cantidad de entradas a comprar. También se ingresa la edad. Se calcula el total a pagar multiplicando el precio por la cantidad. Luego, es necesario descontarle la mitad si es una persona de la 3ra. Edad. Entonces si **ED >=65** (años) a la misma variable **TP** la dividimos 2, de esta manera se le rebaja la mitad.

**Vea** como después de efectuarse este proceso, el flujo se dirige hacia la alternativa de falso. **Es decir, esta Bifurcación realiza un proceso por verdadero y por falso no hace nada**. Continua el diagrama, al final se visualiza el total a pagar.

## Ejercicio 2

Ingrese el sueldo por hora y las horas trabajadas por un empleado. Sólo si el salario neto a recibir por el empleado es menor a $200.00 páguesele por concepto de transporte $50.00 adicional. Visualice el salario a recibir.




Luego de ingresar el sueldo por hora y las horas trabajadas se efectúa el cálculo del salario neto. Si dicho salario es **< 200** se le suma al salario 50. Luego se visualiza presenta **SNR**.

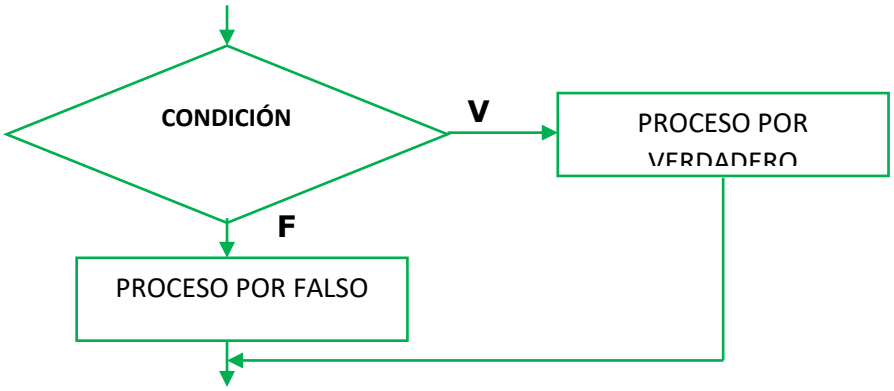
Cuando **SxH** es **1.5** y **HT** vale 40, el salario neto a recibir es 60; entonces la condición es verdadera. **Por eso se le suma 50 y el resultado final es 110**.

Cuando el sueldo por hora es 10 y las horas trabajadas son 40, el salario neto es **400**. Por lo tanto, no recibe bono de transporte porque no es menor que **200**

Entonces la condición **SNR < 200** es falsa. No se efectúa ningún proceso. Al final, ya sea por verdadero o por falso indistintamente siempre se visualiza el salario neto a recibir **SNR**

## Estructuras de Control Dobles

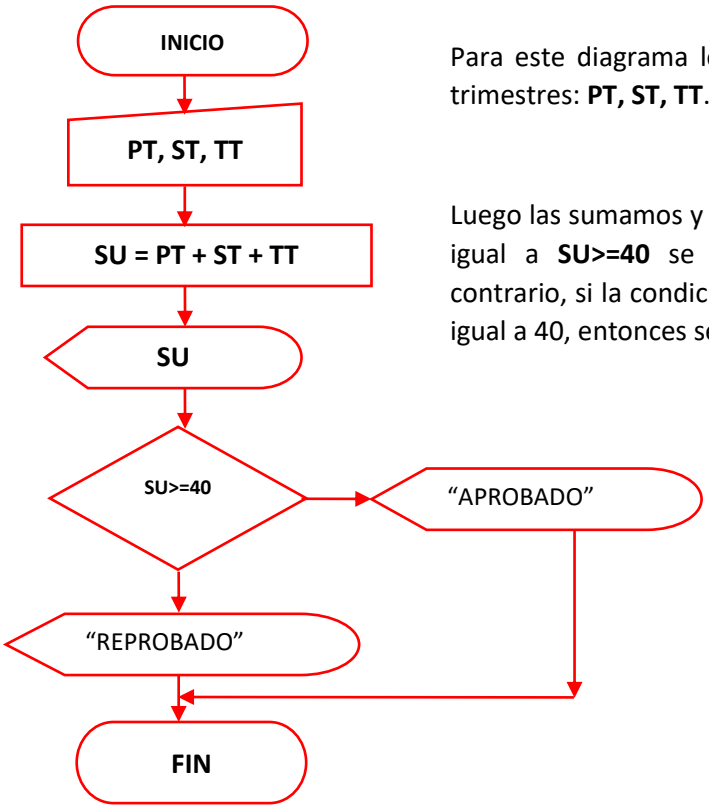
Una Estructura de control es doble, cuando tanto por **verdadero** como por **falso** se realiza la ejecución de alguna **acción** o **proceso matemático**. Las Estructuras de control dobles son de la siguiente forma:




De esta manera, se dice que una Estructura de control es doble cuando se realizan acciones tanto por la alternativa de **verdadero** como por la alternativa de **falso**.

## Ejercicio de Aplicación 03

Ingrese las notas de los 3 trimestres de un estudiante. Visualice su suma. Si dicha suma es mayor o igual a 40 visualice el mensaje "Aprobado". Si la suma es menor a 40, visualice el mensaje "Reprobado".




Para este diagrama leemos las variables equivalentes a los tres trimestres: **PT, ST, TT**.

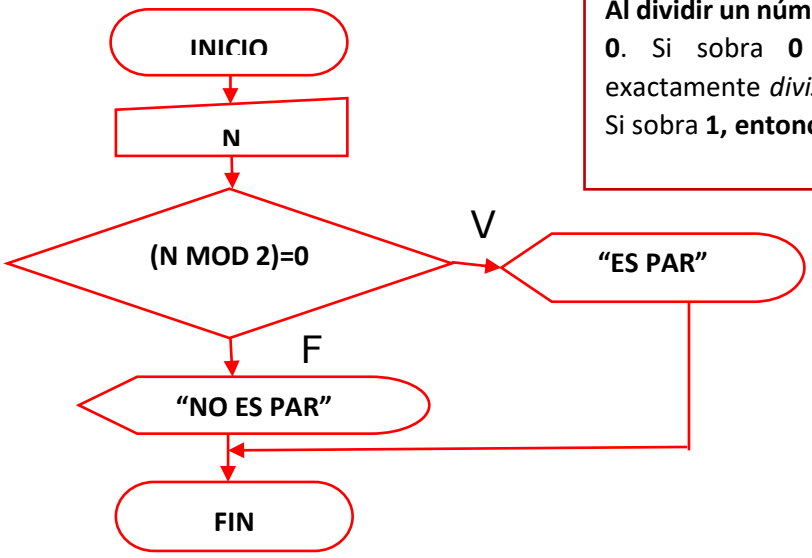
Luego las sumamos y presentamos su suma Si la suma es mayor o igual a **SU>=40** se presenta el mensaje "APROBADO**";** caso contrario, si la condición es falsa, es decir, la suma no es mayor o igual a 40, entonces se presenta el mensaje **"REPROBADO"**

El enunciado del ejemplo dice claramente *"… **si dicha suma es mayor o igual a 40 visualice el mensaje APROBADO; si la suma es menor a 40, el mensaje REPROBADO".*** Nótese cómo la condición **SU >= 40**, satisface el texto del enunciado. Si la suma es mayor o igual a 40 se presenta el **mensaje APROBADO**. Luego dice, si la suma es menor a 40…. Alguien puede pensar que hace falta otra condición: **SU<40**.

Está condición no es necesaria porque su **SU>=40 es falsa**, entonces **SU** es menor que **40**. Está sobreentendido.

## Ejercicio de Aplicación 04

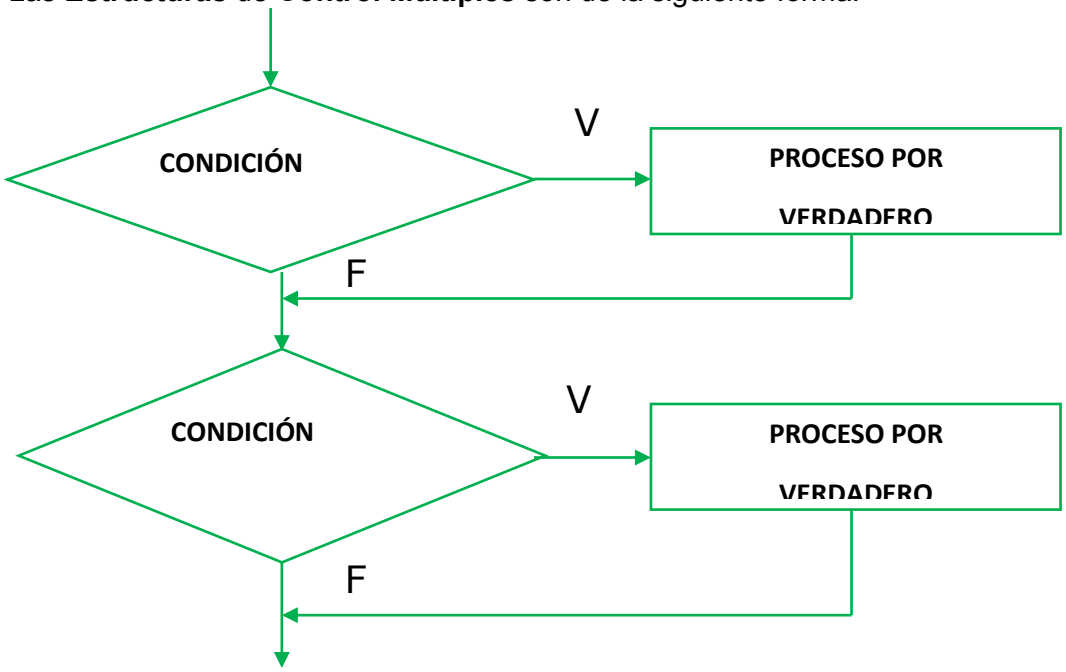
Efectué un diagrama que lea un número. Visualice si es “PAR” o “IMPAR”.




**Al dividir un número para 2 siempre sobra 1 o 0**. Si sobra **0 entonces** el número es exactamente *divisible para 2*, es **decir es par**. Si sobra **1, entonces el número es impar.**

## Estructuras de control múltiple

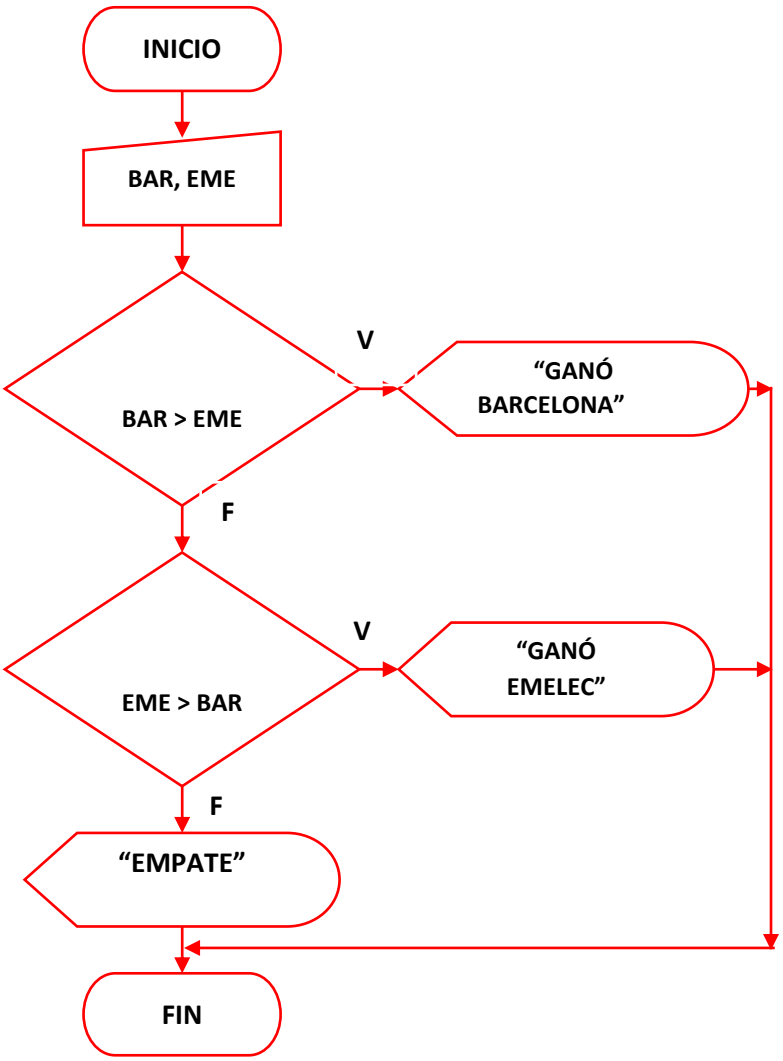
Se dice que una Estructura de Control es Múltiple cuando colocamos varias Estructuras de Control, una a continuación de otra.
Las **Estructuras** de **Control Múltiples** son de la siguiente forma:




Donde podemos colocar una determinada cantidad de estructuras, siempre una a continuación de otra.

## Ejercicio 1

Realice un diagrama que lea el marcador de un clásico del astillero, es decir, cuantos goles marco el equipo de BARCELONA y cuantos anoto el equipo del EMELEC. Visualice el mensaje con el nombre del equipo ganador o si es que hubo empate.




Para este diagrama ingresamos los goles tanto de BARCELONA como de EMELEC. Si la condición **BAR > EME** es verdadera, significa que BARCELONA consiguió un mayor número de goles que EMELEC. Entonces gana BARCELONA.

Si la condición es falsa, preguntamos si **EME > BAR**. Si es verdadera gana EMELEC. Sin embargo, si esta nueva condición también es falsa, entonces significa que hubo un empate.

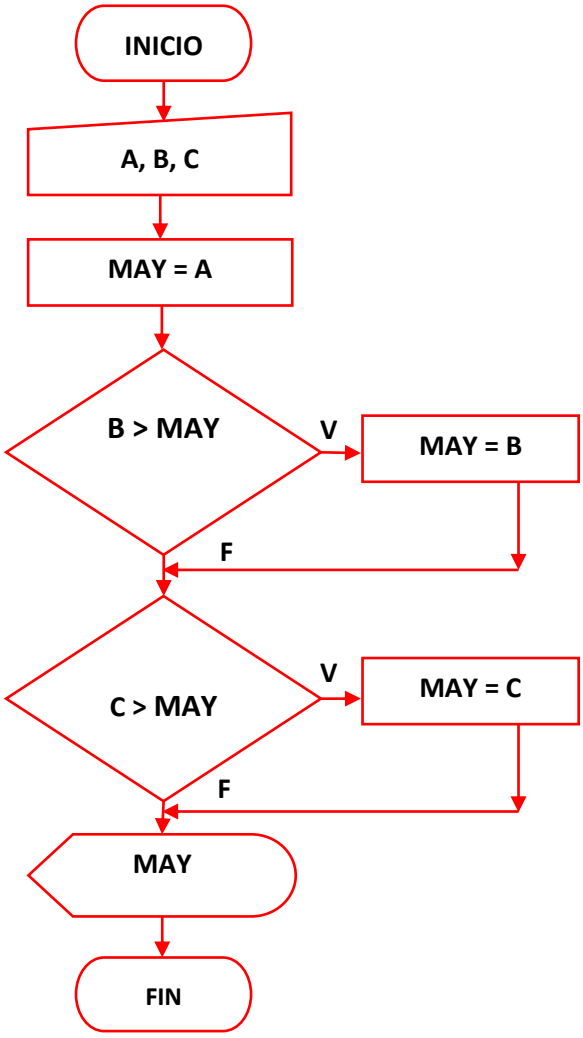
Analice. Si BAR > EME es falsa y EME > BAR también es falso, entonces BAR y EME son iguales.

| BAR | EME | MENSAJE |
|---|---|---|
| 2 | 1 | *GANÓ BARCELONA* |
| 1 | 3 | *GANÓ EMELEC* |
| 0 | 0 | *EMPATE* |

## Ejercicio 2

Realice un diagrama de flujo que lea tres números. Visualice al mayor de ellos.

Para facilitar la búsqueda del número mayor entre tres, en este diagrama usamos un pequeño artificio. Ingresamos los tres valores y luego asumimos que el mayor de todos es el primero. De esta manera, almacenamos en la variable **MAY** el valor de la variable **A**. Luego preguntamos si B es mayor que **MAY** (o sea A). Si la condición es verdadera, entonces el nuevo mayor es **B**. Así **MAY = B**. Ahora preguntamos si **C** es mayor a MAY. Ya en ese momento, **MAY** contiene el mayor valor entre **A** y **B**. Si **C > MAY** es verdadero, entonces almacenamos **C** en **MAY**.




Este es un algoritmo muy usado para encontrar el mayor de varios números. Se asume que el primer valor es el mayor, y luego se hacen sucesivas comparaciones con los demás valores. En el momento que uno de estos valores sea menor, se intercambian los valores. Veamos la prueba:

| A | B | C | D |
|---|---|---|---|
| 3 | 5 | 9 | 3<br>5<br>9 |

# UNIDAD 4

SCRATCH

# PROGRAMACIÓN SCRATCH

## PROGRAMACION CON SCRATCH


**Scratch** es un lenguaje de programación creado por el MIT y especialmente diseñado para que todo el mundo pueda iniciarse en el mundo de la programación. Su principal característica consiste en que permite el desarrollo de habilidades mentales mediante el aprendizaje de la programación sin tener conocimientos básicos.

El nombre proviene de la palabra: "**Scratching**" que, en los lenguajes de programación, significa aquellos trozos de código que pueden ser reutilizados, fácilmente combinables y adaptados para nuevos usos (**es software libre**)y de fácil uso, se la puede descargar a nuestro ordenador para los diferentes sistemas operativos.

Las acciones y comportamientos están divididas en categorías y son estas:


**Movimiento**: Mover y girar un objeto por la pantalla.


**Apariencia**: Cambiar la visualización del objeto: el fondo, hacerlo más grande o pequeño, etc.


**Sonido**: Hacer sonar secuencias de audio.


**Eventos**: Maneadores de eventos que "disparan" determinadas acciones en un bloque.


**Control**: Condicionales: repetir, condiciones, detener.


**Sensores**: Los objetos o "sprites" pueden interaccionar con el entorno o con elementos creados por el usuario como un robot lego, por ejemplo.


**Operadores:** operadores matemáticos, generadores aleatorios de números, cooperadores de posiciones.


**Variables**: Crear variables y su asignación en el programa.


**Mis bloques:** Bloques propios y controladores de aparatos externos.

**Utilidad.**

- Es perfecto para introducirse en la programación.
- Pudiendo ser descargados y utilizados por otras personas.
- Permite compartir los proyectos a través del web, se pueden descargar y utilizar.
- Permite el desarrollo de los procesos de pensamientos y habilidades mentales en los educandos.

## ¿Por qué Scratch?

La programación es el nuevo lenguaje que todos necesitamos conocer si queremos tener una buena comprensión del mundo actual y sobre todo del que viene, así como buenas oportunidades laborales. se calcula que cerca del 50% de los puestos de trabajo que conocemos hoy en día desaparecerán y en buena medida serán sustituidos por la industria del software y a la robótica. De este modo Scratch se convierte en una gran herramienta para comprender los conceptos y la lógica de la programación. Además, lo hace abordando su aprendizaje desde un punto de vista lúdico para evitar el rechazo inicial que para muchas personas suponen los entornos de programación más clásicos.

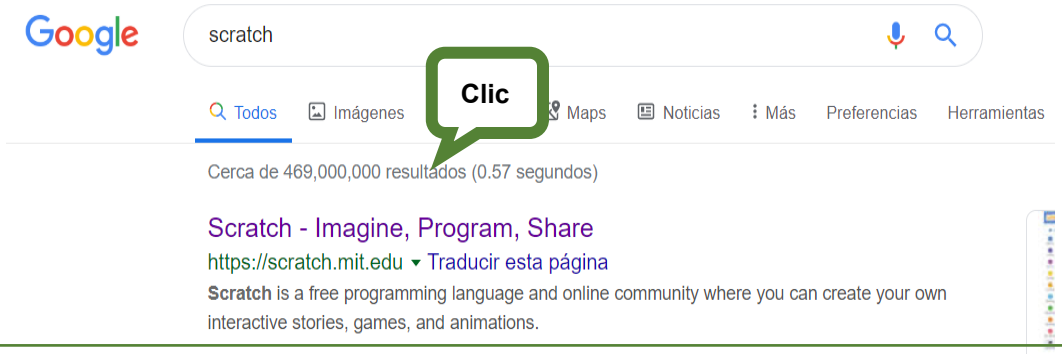
## Ventajas para el desarrollo del niño

Si hay un entorno en el que **Scratch** esta especialmente indicado, es en el de la enseñanza de la programación a los niños, por ese componente lúdico del que hemos hablado antes. Podríamos agrupar una serie de ventajas que su uso proporciona a los niños:

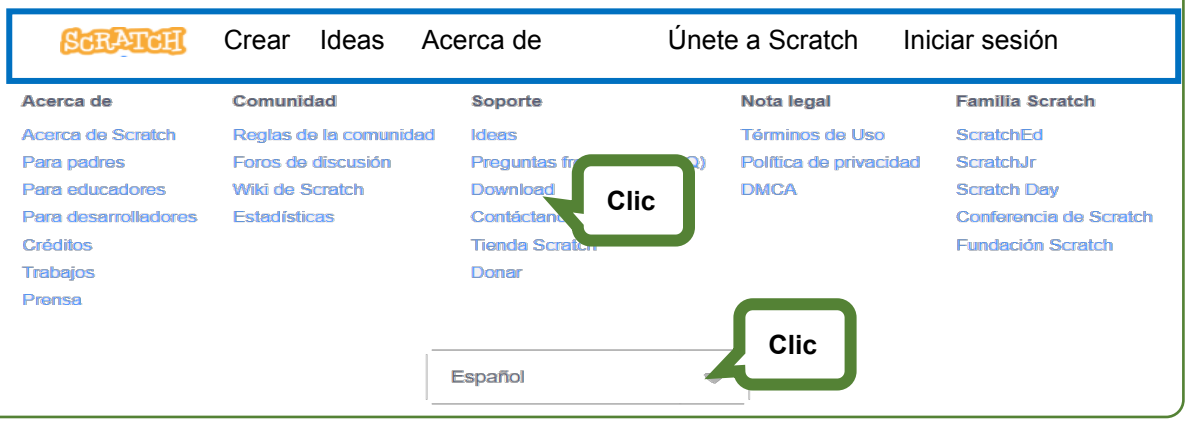
- Desarrollar el pensamiento lógico.
- Aprender los fundamentos de la programación.
- Usar distintos medios: sonido, imagen, texto, gráfico.
- Desarrollar la capacidad de poner en duda las ideas de uno mismo.
- Trabajar cada cual a su ritmo en función de sus propias competencias.
- Desarrollar el hábito de hacer auto diagnosis con respecto a su trabajo.
- Tener la posibilidad de obtener resultados complejos a partir de ideas simples.
- Desarrollar métodos para solucionar problemas de manera metódica y ordenada.
- Posibilitar el aprendizaje colaborativo a través del intercambio de conocimiento.
- Aprender y asumir conceptos matemáticos: coordenadas, variables, algoritmos, aleatoriedad.

## Descargar Scratch

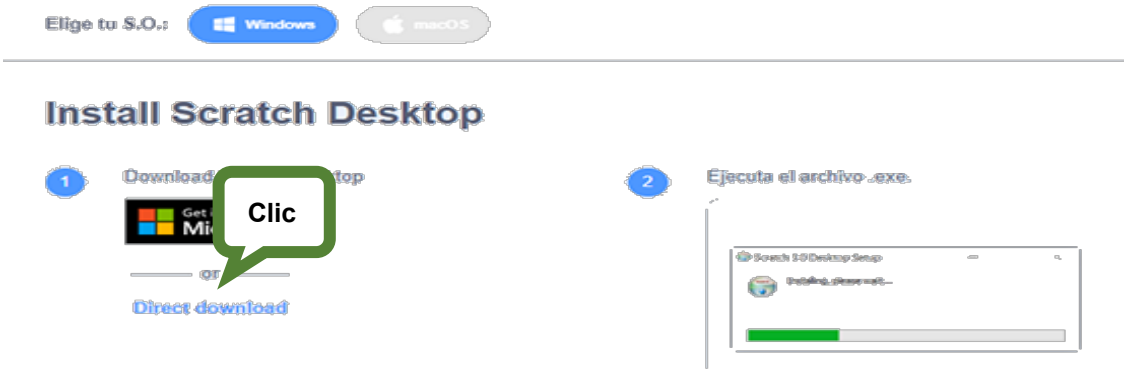
En el navegador de su preferencia digite **Scratch**:




Después de dar clic, se presenta la siguiente ilustración y de clic en:



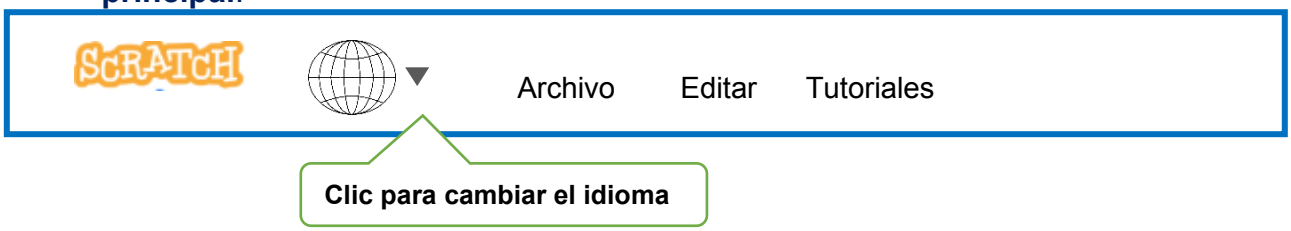

La siguiente página, es para descargar la aplicación, de clic en:



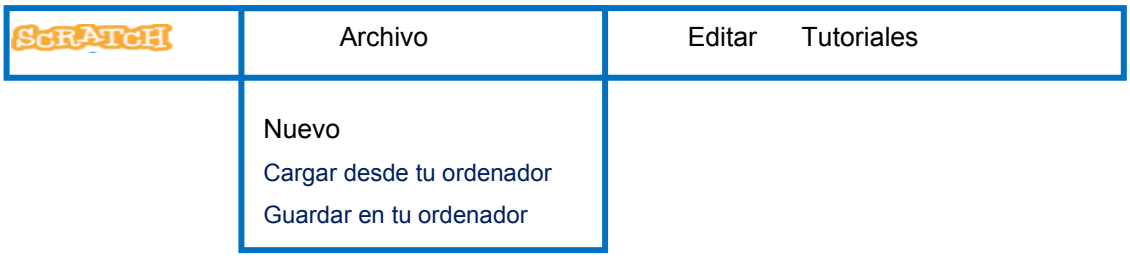

A continuación, comienza la descarga de la aplicación y comience con la instalación:

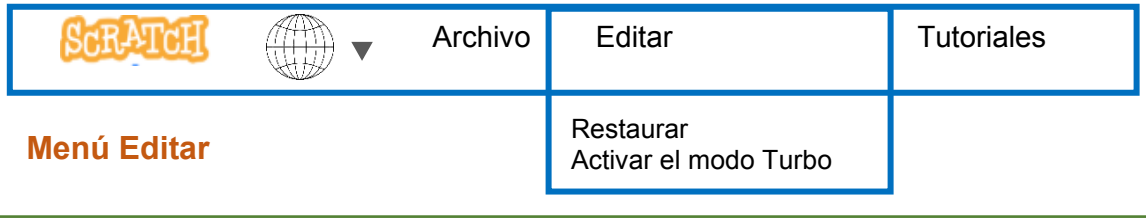
A continuación, se visualiza el **Entorno** de **Scratch**:

Lo primero que se hará es configurar el **idioma,** en la **barra de menú principal**:



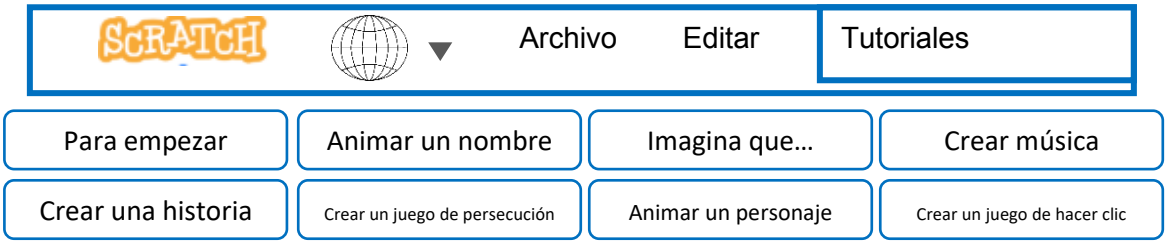

**Barra de menú Principal:** Ubicada en la parte superior y color azul, Menú **Archivo**, se visualiza **Nuevo**, **Cargar desde tu ordenador** y **Guardar en tu ordenador**.



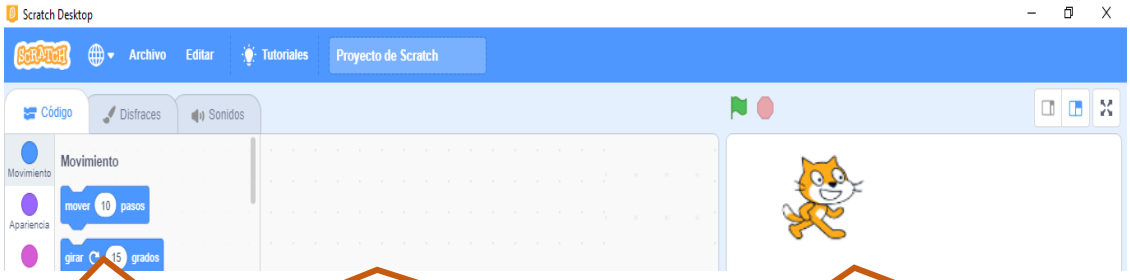

**Menú Editar**

**Menú Tutoriales** (visualiza una gama de ejemplos a seguir)




**Asignar nombre al proyecto**

## Área de trabajo

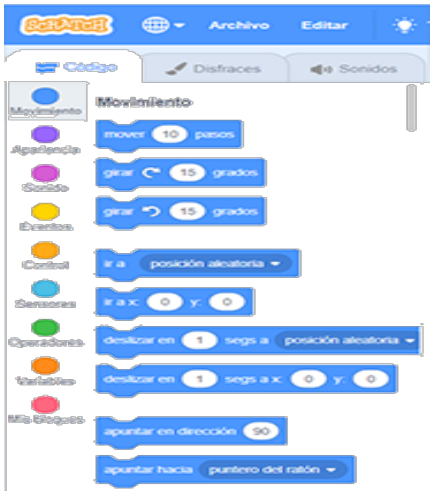



**Paleta de Bloques:** Se visualizan el **Código** de la programación, **Disfraces y Sonidos.**

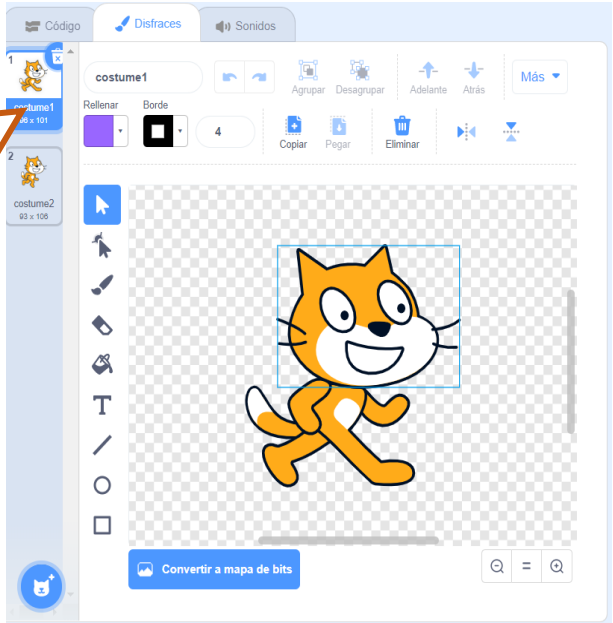
**Área de código:** espacio donde se programa en bloque, arrastramos los códigos para crear la programación que desee.

**Escenario:** Lugar donde aparecen y se mueven todos nuestros **objetos** del proyecto.

**Paleta de Bloques**: Se presentan a la izquierda nuestra caja de instrucciones. Son los bloques para crear los scripts o programas.
Tenemos 9 cajas diferentes de piezas, para distintas acciones, cada una de un color diferente: **Movimiento, Apariencia, Sonido, Eventos, Control, Sensores, Operadores, Variables, Mis Bloques**.

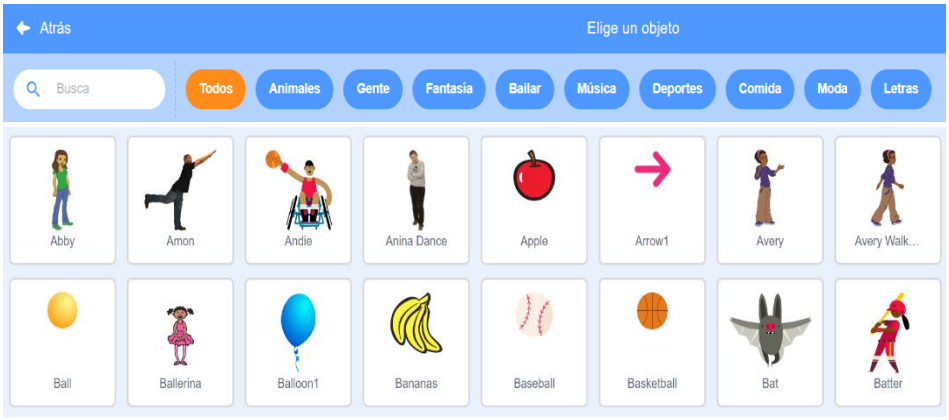



**Disfraces:** pestaña la cual contiene diferentes disfraces también se los conoce como **objetos**, **botones** permitir buscar o incluso crear nuevos actores para nuestra acción.

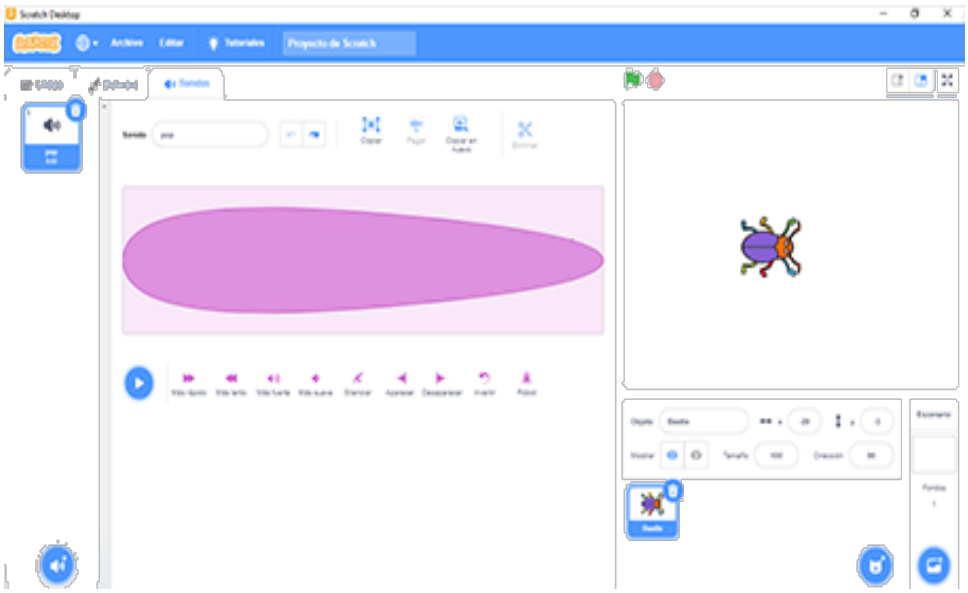



Para poder elegir un objeto diferente (**disfraz**) el cual se lo puede obtener dándole clic al botón que está en la parte derecha inferior del escenario su icono es el siguiente:

**Lista de objetos**: Aquí puede escoger una de las siguientes ilustraciones (**objetos**), que son de varios tipos, haga clic para seleccionar y editar un objeto.




**Sonido:** Aquí puede seleccionar el sonido del objeto, también se puede cargar una música o una parte de ella.

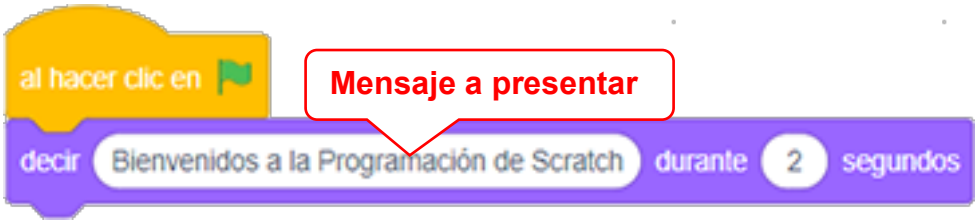


**En la siguiente práctica vamos a presentar un mensaje, para lo cual debe crear un nuevo proyecto:**

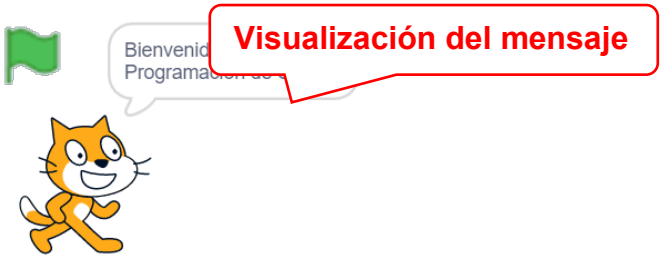
Primero debemos empezar en **EVENTOS** arrastrar: **hacer clic en**


Luego deseamos que se presente un mensaje


Entonces tenemos:

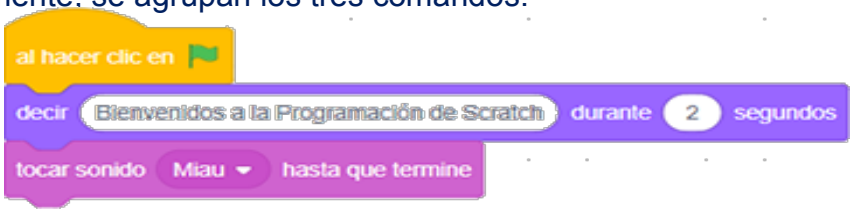



Damos clic en




Vamos agregar sonido a nuestra presentación.


Finalmente, se agrupan los tres comandos:



**A continuación, vamos a realizar un nuevo proyecto, que se saluden dos personajes, para lo cual debe agregar otro objeto:**


Recuerde que siempre debe empezar con el comando


Luego vamos a mostrar dos mensajes:


Vamos agregar un nuevo objeto:

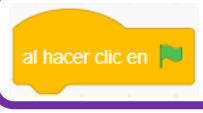


Agregado el objeto **Dog1**, luego arrastre los siguientes comandos


Ejecutamos el proyecto dando clic en

**Nuevo proyecto**: Arrastre uno de los comandos hacia el **Área de código** y el conforme presione el código va generando una acción que se visualiza en el **Escenario**, ejemplo: vamos hacer que **Scratch** realice un giro.


**Paleta de Bloques** | **Área de código** | **Escenario**

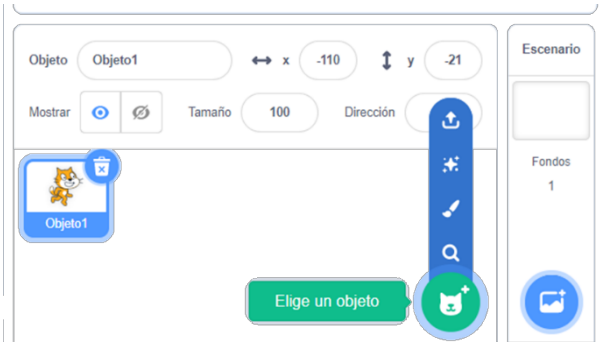
**NOTA**  Siempre debe empezar con este comando

 **Bandera verde:** es utilizado para ejecutar una acción del código.

 **Botón rosado:** sirve para detener la acción.

 **Control o pantalla completa:** permite ver solo el área del escenario la cual es donde aparece el objeto listo para ser ejecutado.

**Cambiar un objeto (imagen):** esto permite cambiar la imagen que viene por defecto por otro de nuestro agrado, de clic en la parte inferior derecha:

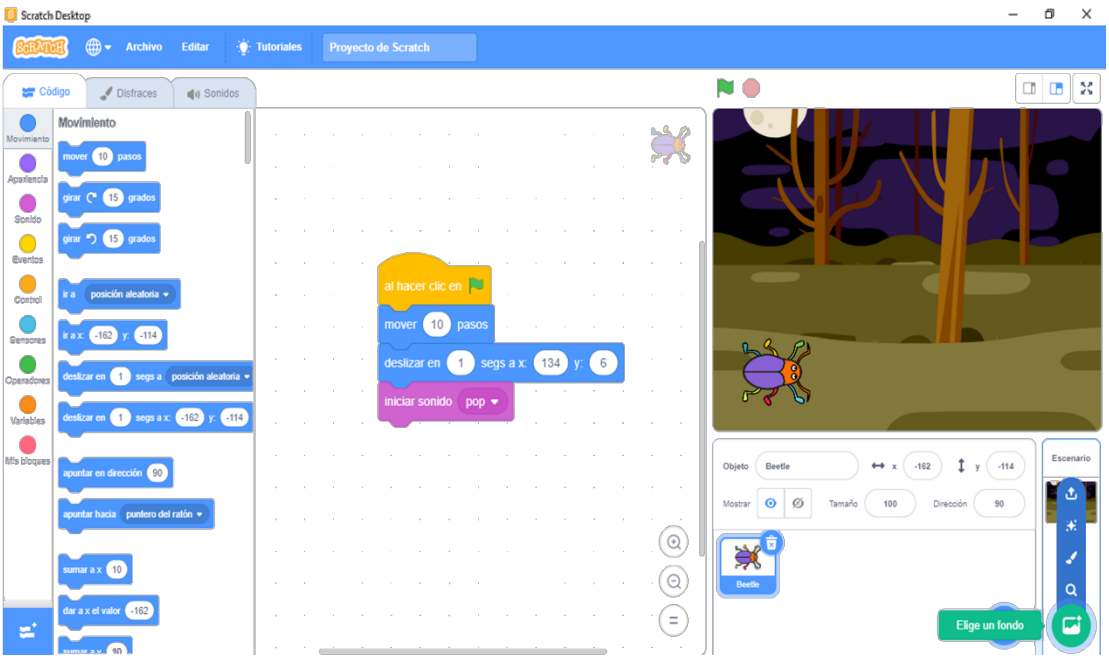




**Cambiar fondo (escenario):** esta operación permite cambiar el fondo del escenario, de clic en la parte inferior derecha:


## Ejercicio de Aplicación 1

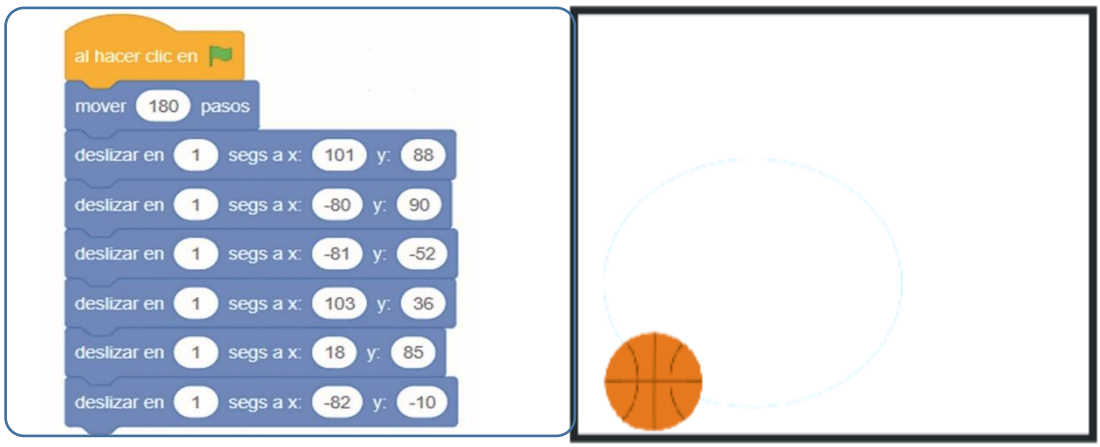
Hacia el área de código, arrastre los siguientes comandos:



Para ejecutar de clic en la siguiente ilustración:  como se observa el objeto se desliza de un lugar a otro y finalmente emite un sonido.

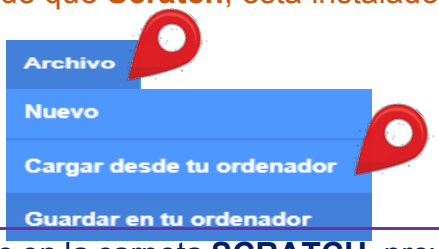
El siguiente proyecto, permite que un balón de básquet siga la parte interna de un cuadrado, pestaña **Fondo** y desarrolle de acuerdo a la siguiente ilustración:


Dentro del área de código, arrastre los siguientes comandos:

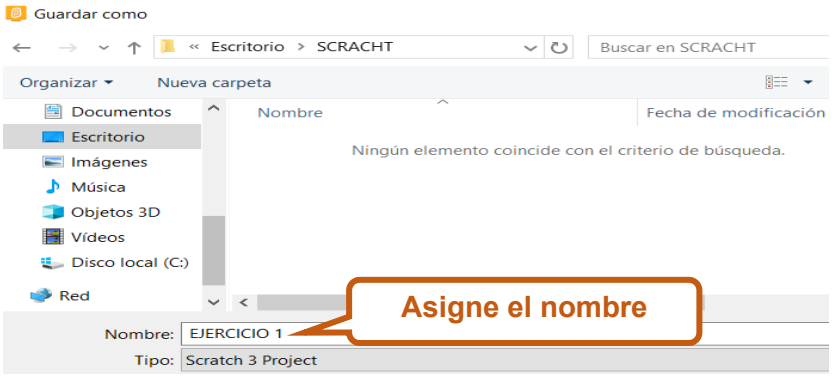


## Guardar un Proyecto, en nuestro equipo.

Esta operación consiste en almacenar el proyecto en algún lugar dentro del computador. Recuerde que **Scratch**, esta instalado en su computador.



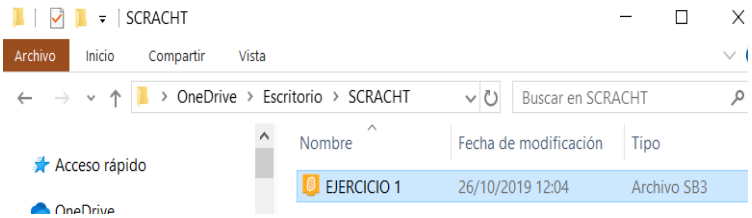

Almacene el proyecto en la carpeta **SCRATCH**, previamente creada, asigne el nombre **EJERCICIO 1**




En la barra de Menú, se aprecia el nombre del proyecto **EJERCICIO 1**


Ahora cierre la aplicación de Scratch, de clic en la parte superior derecha y se presente el siguiente mensaje:

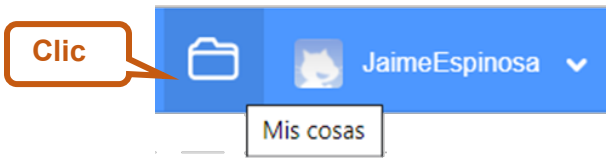

Finalmente, vamos a la carpeta de **SCRATCH**, y damos doble clic en el siguiente icono automáticamente se carga el proyecto



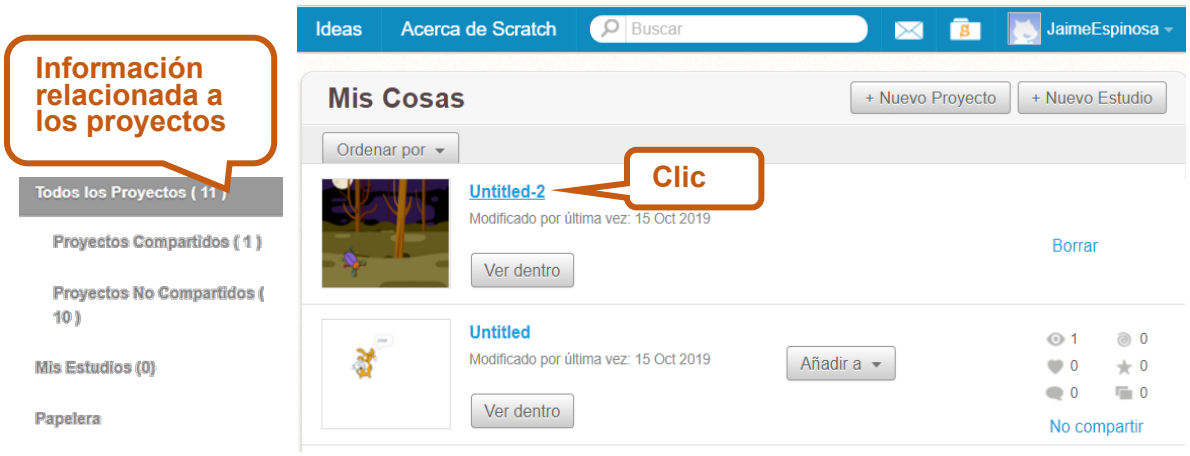
## Guardar Proyecto en la Nube de SCRATCH

Lo primero que se debe realizar es ir al navegador y digitar la siguiente dirección electrónica: **http://scratch.mit.edu/** registrarnos como usuario en línea.


Después de registrarse, y desarrollar ejercicios o proyectos en línea se almacenan en la barra de Menú en la opción **Mis cosas**

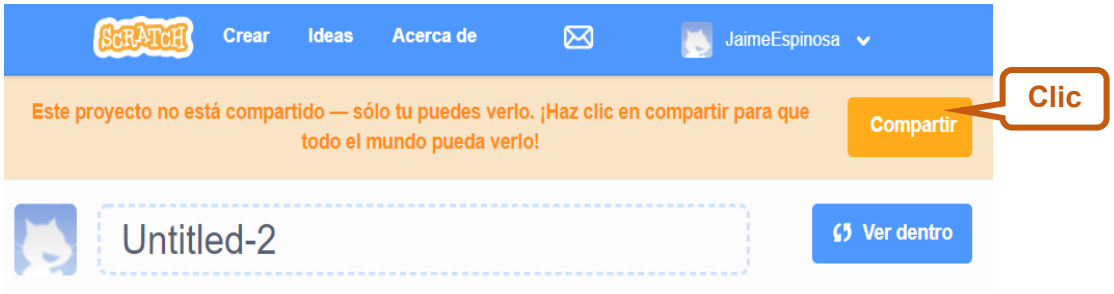



En nuestro caso, hemos desarrollado varios ejercicios como se puede apreciar:

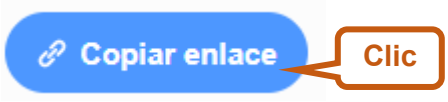



Después de dar clic, en el nombre del proyecto, se carga el mismo, **recuerde que está trabajando en línea, es decir, trabaja con un navegador.**

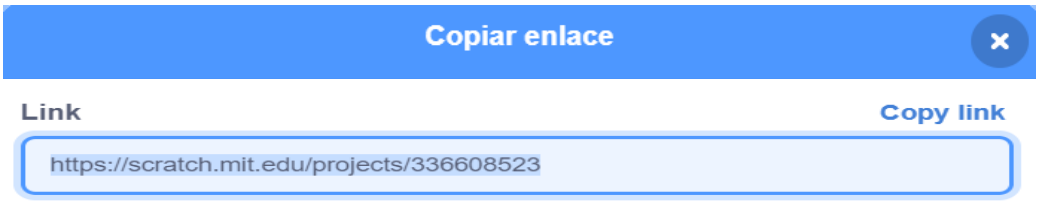
La siguiente ilustración, muestra el botón **COMPARTIR**



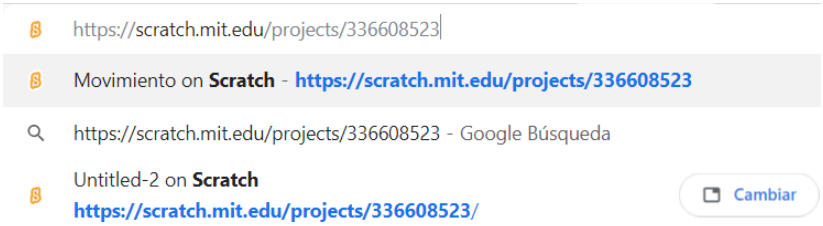

Después de dar clic en **COMPARTIR**, se presenta la ilustración y damos clic en **Compartir enlace:**




Una vez que, de clic en **Compartir enlace,** de clic en **Copy link,** y automáticamente se copia el **Link**, ahora lo puede compartir




Vamos al navegador y se pega el **link**




Finalmente se visualiza el proyecto para ser ejecutado

## Utilizar Sonido

Para que nuestro proyecto tenga una mejor presentación, vamos agregar un **SONIDO**, que se escuchará cuando se ejecute, escoja uno de los sonidos que vienen por predeterminado:

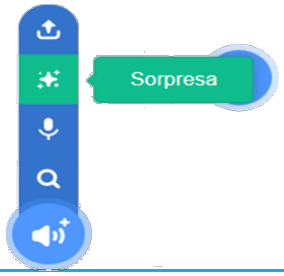


Al dar clic en **Elige un sonido**, se presentan los audios que vienen por predeterminados:



Si desea, puede realizar una grabación para ser agregada al proyecto:

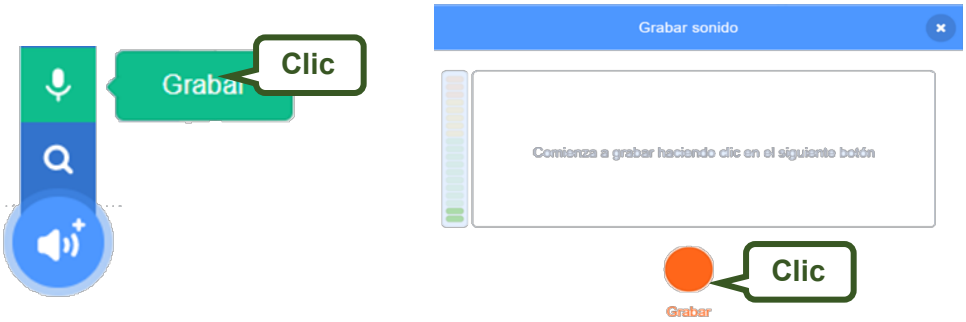

En esta opción, viene un sonido de tipo sorpresa.




Esta opción se la utiliza para cargar sonido desde algún lugar de su computador

## Grabar un Sonido

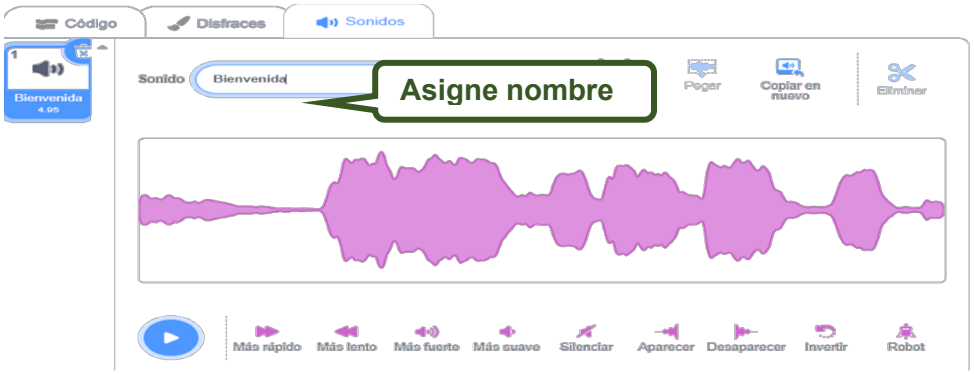
Vamos agregar un sonido a nuestro proyecto,




Después de dar clic en grabar, empieza a grabar el audio y se presenta la siguiente imagen:

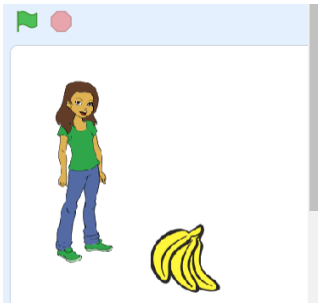

Al terminar la grabación se muestra la imagen para reproducir y Guardar el audio.


Finalmente, se graba el audio y se observan los botones de navegación para su reproducción y asigne nombre



Para este nuevo proyecto vamos a utilizar el ingreso de texto y se va escuchar un audio, personalizado.

Creamos un **Nuevo Proyecto** y añadimos dos objetos de acuerdo a la siguiente ilustración:



Ahora vamos a programar en bloques:


Asignar una variable, para esto sigua los siguientes pasos:


Nos toca añadir audio, prepárese para hacerlo:

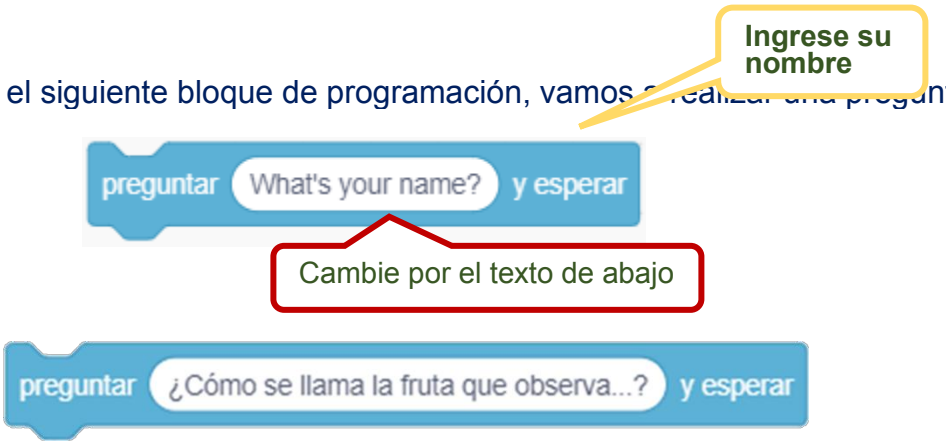
En la siguiente imagen de clic en **Guardar**


Después de dar clic en guardar, aparecerá la siguiente ilustración:


El siguiente paso es cargar el comando sonido y escoger **Bienvenida**


Ejecutamos el proyecto y pide que ingrese un dato y después se escucha un audio que se agregó.

Para el siguiente bloque de programación, vamos a realizar una pregunta:

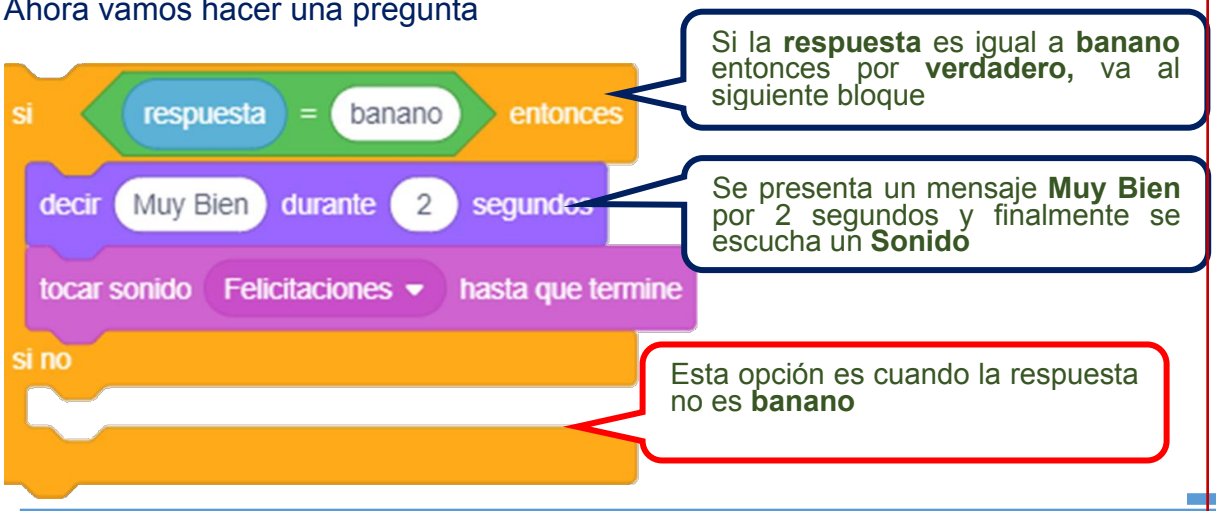



Después de realizar la pregunta:
**¿COMO SE LLAMA LA FRUTA QUE OBSERVA…?**

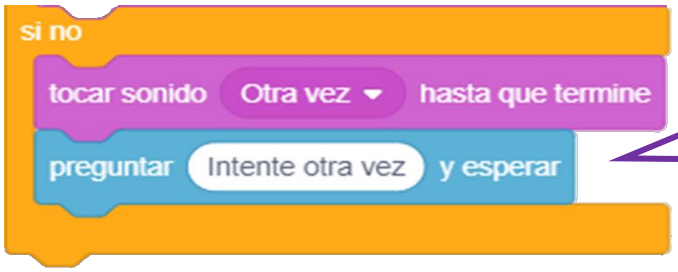

Vamos a escribir la palabra **banano**, que será almacena como **respuesta**

Ahora vamos a **repetir hasta que** la **respuesta** sea **igual** a **banano**


Si la **respuesta** no es **igual** a **banano**, entonces tenemos **dos opciones** una por **verdadero** y otra **por falso.**
Ahora vamos hacer una pregunta



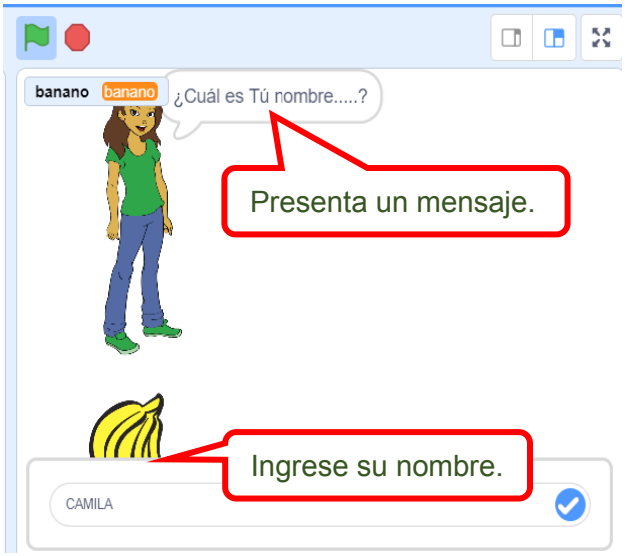
A continuación, realizamos estas acciones por **falso**




Se escucha el **audio** y realizamos una **pregunta,** y se presenta un mensaje.


Finalmente agregamos un mensaje.

Ejecutamos el programa y se presenta la siguiente ilustración:



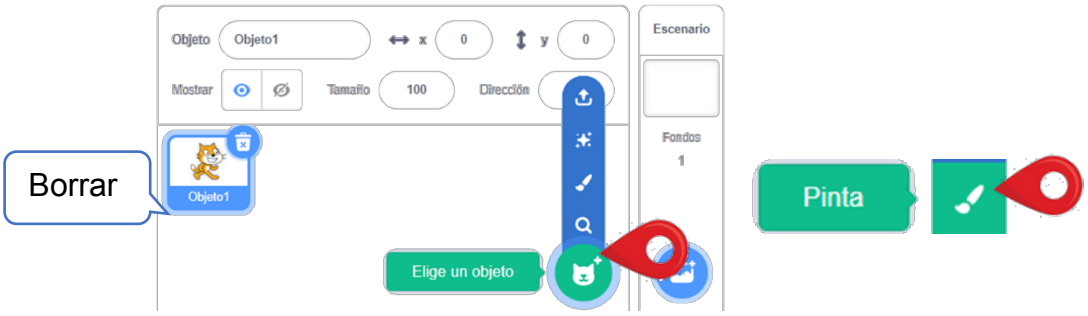



El siguiente ejercicio, tenemos la apariencia de una carretera infinita, vamos a crear un nuevo proyecto:

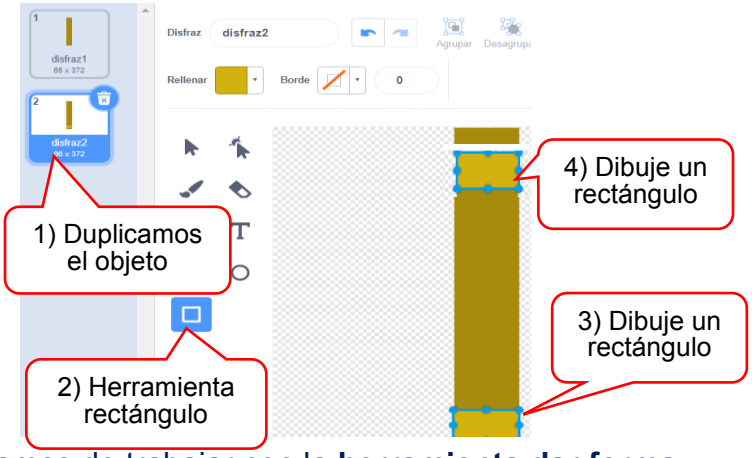
**Archivo**

**Nuevo**

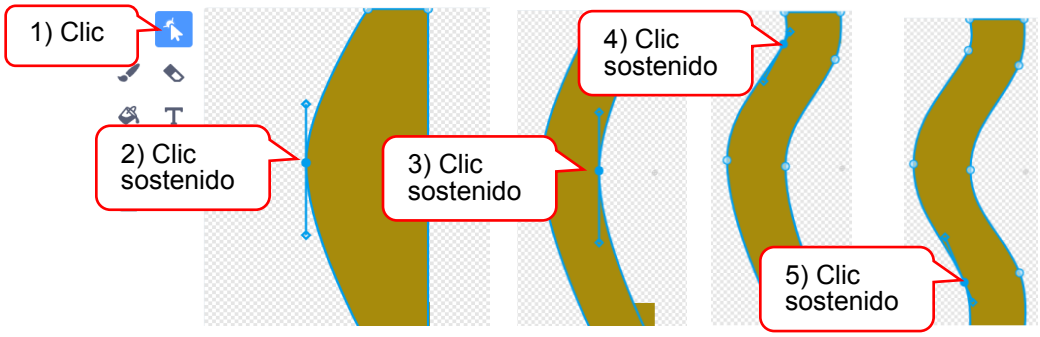
Luego vamos Disfraces donde creamos un nuevo objeto:

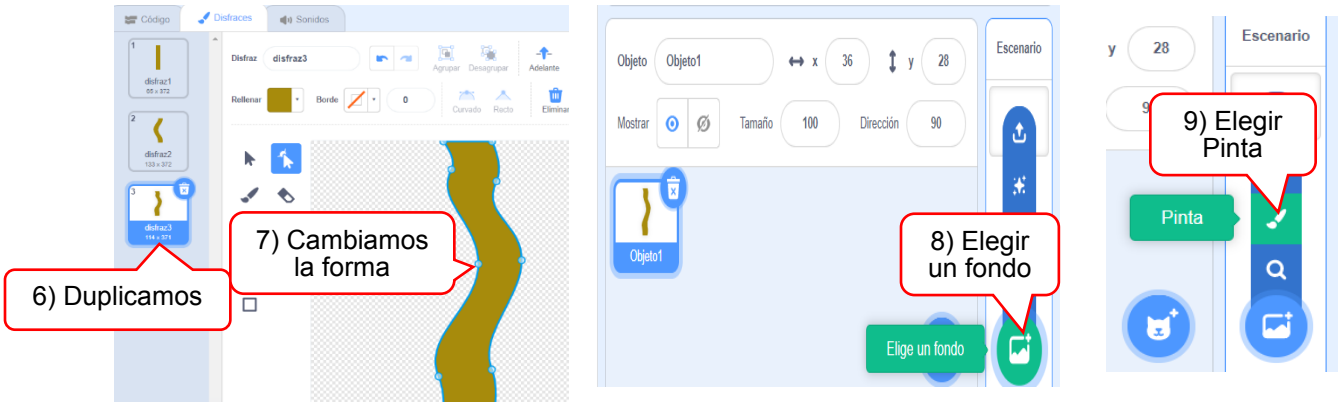


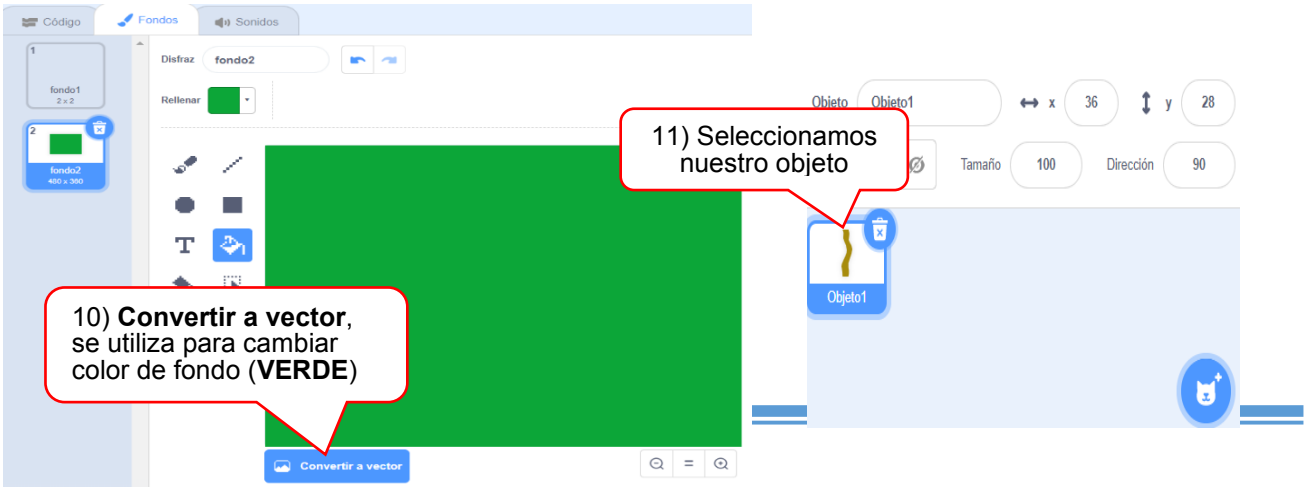





Ahora vamos de trabajar con la **herramienta dar forma**



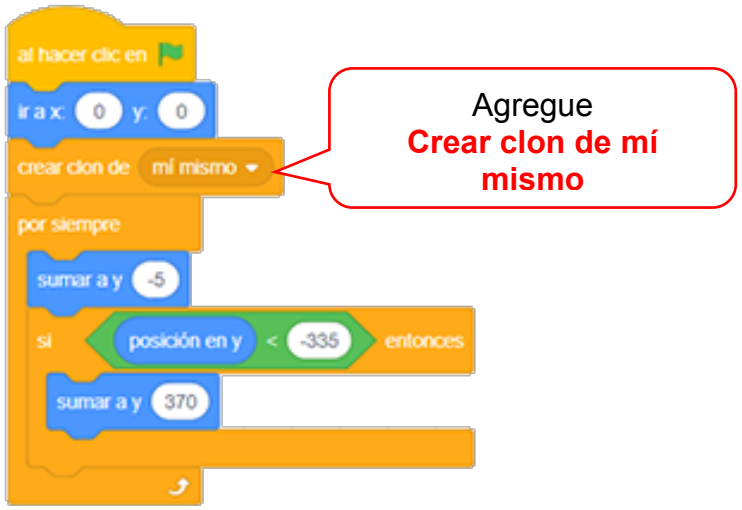



A continuación, vamos a programar, de clic en Código


Ejecute el proyecto con el icono

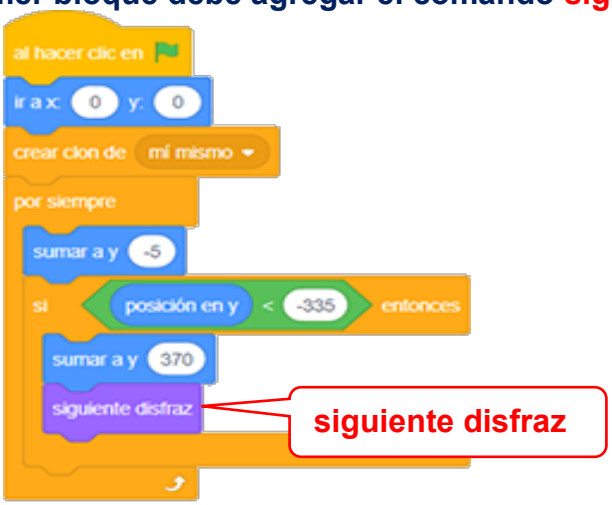

Es decir, vuelve hacia abajo y se traslada arriba, no existe continuidad en la pista, arrastre un clon del mismo objeto y realice lo siguiente:



**Después de agregar un clon, y se debe Duplicar el bloque:**


**Después de duplicar el bloque, agregue el comando:**
**Al comenzar como clon**


**Finalmente, al primer bloque debe agregar el comando siguiente disfraz**



**Programación en Scratch,** consiste en una serie de pasos, secuencias lógicas y ordenas a seguir, arrastrando los objetos para conformar un bloque, que componen un programa, para lo cual es necesario conocer ciertas definiciones.

## Variables

Una variable es un ***casillero de memoria*** en el cual podemos almacenar datos. Estos datos pueden variar dependiendo de la ejecución del programa. Toda variable debe tener un nombre. Los nombres de las variables deben cumplir con las siguientes reglas:

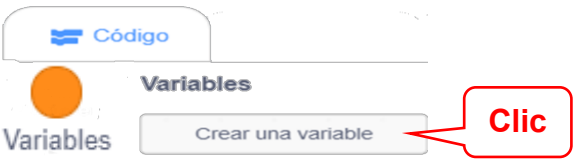
Deben de empezar siempre con una ***letra***, (Mayúscula o Minúscula) y pueden ser una ***combinación*** de ***letras*** y ***números***. Por ejemplo, nombres correctos de variables son: SUMA, XX, M1, M2, XYZ123, SUELDO, entre otros.

**No** pueden tener ***espacios en blanco,*** ni símbolos especiales intermedios. Para nombres de variables largos, use abreviaturas (sin puntos). Por ejemplo: incorrecto es usar SUELDO POR HORA, correcto sería SUELXHOR.

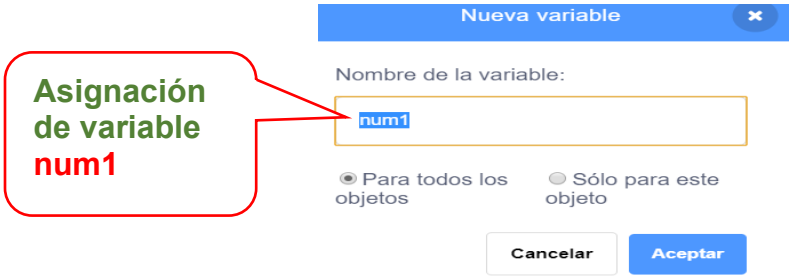
Deben ser nombres ***cortos*** **y** significativos. Evite usar nombres largos y difíciles. Por ejemplo, en lugar de usar la variable PROMEDIO use PROM.

Cada nombre de variable debe ser ***único***, es decir, no puede existir otra variable con el mismo nombre.

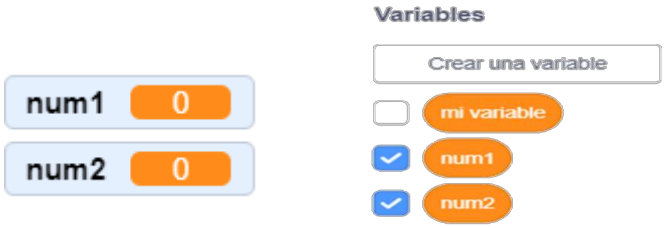
**Práctica,** el siguiente proyecto realiza la **suma de dos valores**.




**Después de dar clic en Crear una variable, se presenta la siguiente ilustración donde se asigna la variable a utilizar para la programación.**



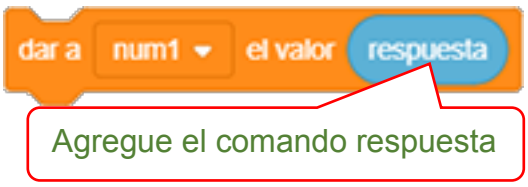
Repita la operación anterior y asigne otra variable para la operación de la suma, es decir, tenemos **num1** y **num2:**




Iniciamos con la programación y agregamos una pregunta:


Luego agregue, el siguiente comando y selecciónela primera variable **num1**


El siguiente paso es cambiar **el valor 0**, por el comando **respuesta**, donde se almacena el número que se ingresa:

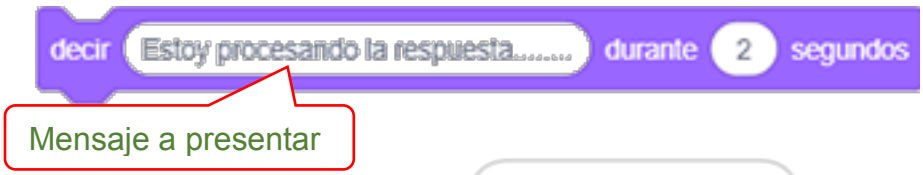



Entonces nos queda así:

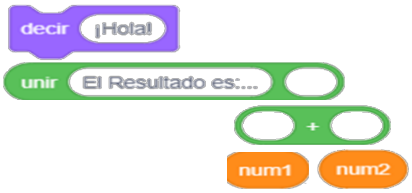
Realizamos la misma operación para el ingreso del segundo número, duplicamos los comandos:

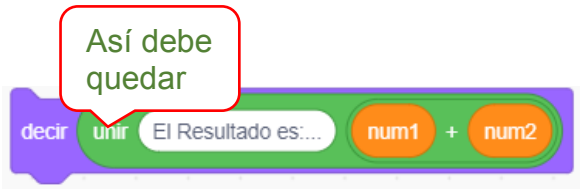

Recuerda que los números que se ingresan se almacenan en el comando respuesta para las dos variables:


Agrego un comando que presente el siguiente mensaje por dos segundos:

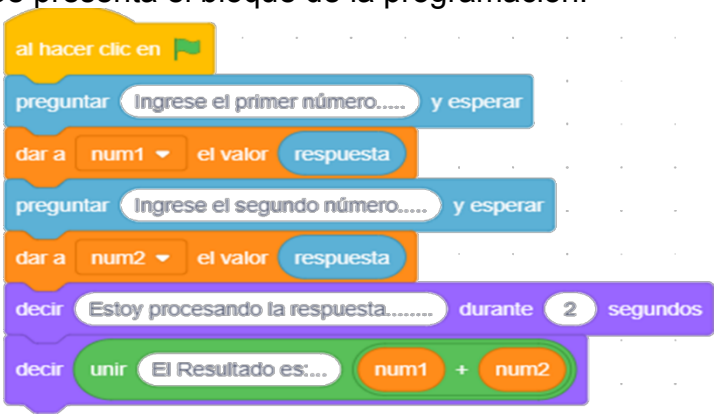



Ahora vamos a presentar un mensaje y realizamos la operación matemática:



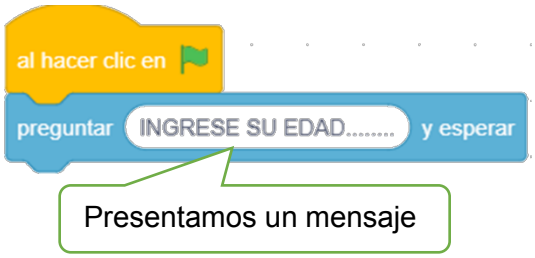

Se presenta el bloque de la programación:

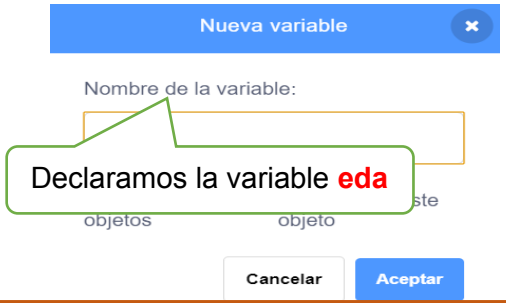



Ahora ejecutamos el programa:



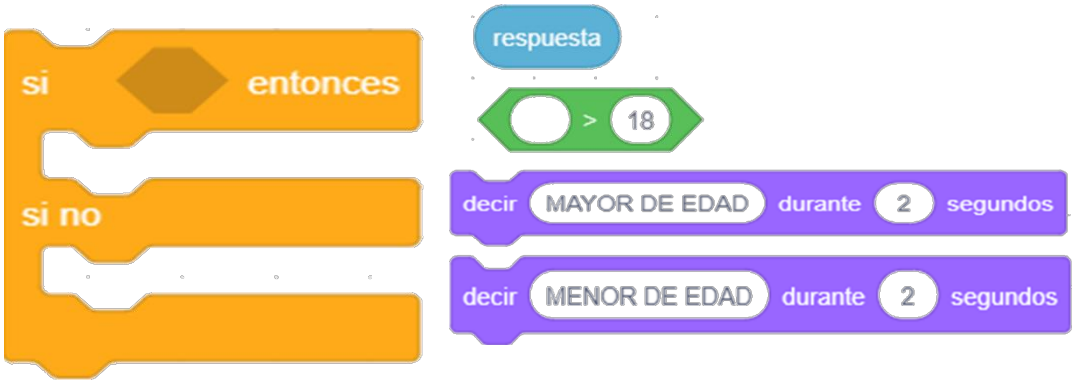

En el siguiente ejercicio vamos a utilizar una **condición**, también conocida como una pregunta que tiene dos alternativas una por verdadero y otra por falso; se solicita ingresar la edad de una persona y presente un mensaje **Mayor de Edad** o **Menor de Edad**, según corresponda. **Recuerde** que toda persona es mayor de edad a partir de los 18 años.




En el siguiente comando seleccionamos la variable **eda** que se almacena en **respuesta**


Realizamos una condición con dos respuestas, por verdadero debe presentar un mensaje **MAYOR DE EDAD**, por falso **MENOR DE EDAD**




Tanto por **verdadero** como por **falso** vamos a presentar un mensaje.

## Práctica NN

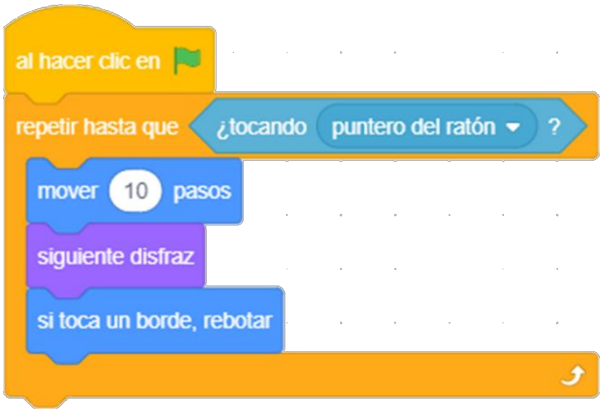
Para el siguiente ejercicio utilice el objeto que se encuentra a continuación, lo vamos hacer girar por todo el escenario, se detendrá hasta tocarlo con el puntero del mouse:

Recuerde que debemos empezar con el comando **al hacer clic en la bandera,** vamos a **repetir hasta que** tocando **puntero del ratón**


Después de repetir, debe **moverse 10 pasos**, conforme va girando, también se cambia de disfraz, finalmente **si toca un borde, rebota.**


El bloque se visualiza de acuerdo a la siguiente imagen:

‘’



## *Descubre tu próxima lectura*

Si quieres formar parte de nuestra comunidad,
regístrate en **https://www.grupocompas.org/suscribirse**
y recibirás recomendaciones y capacitación

compAs

Grupo de capacitación e investigación pedagógica

@grupocompas.ec
compasacademico@icloud.com


@grupocompas.ec
compasacademico@icloud.com

ISBN: 978-9942-33-295-0


@grupocompas.ec
compasacademico@icloud.com