Universidad Tecnológica de Pereira

Mision TIC 2022

El futuro digital es de todos

MinTIC

Algoritmos

Conceptualización, representación y codificación

Hechos QUE CONECTAN

Contenido

- Lógica y Algoritmos
- Algoritmos Informales
- Algoritmos Computacionales
  - Variables
  - Estructuras
- Conceptualización, representación y codificación

Programación vs Lógica de Programación

• **Programación:** Involucra el conocimiento de técnicas e instrucciones de un determinado lenguaje a través de las cuales se nos hace sencillo lograr que el computador obtenga unos resultados mucho más rápido que nosotros.

• **Lógica de Programación:** Conceptos que nos permiten diseñar, en términos generales, la solución a problemas que pueden llegar a ser implementados a través de un computador.

Lógica de Programación

- No exige ***ningún*** conocimiento previo de computadores ni de tecnología en general.

- Tampoco exige la presencia de algún lenguaje de programación específico.

- Dominar un lenguaje de programación, sólo después que usted maneje bien los conceptos de lógica de programación, le permitirá implementar y ver convertida en realidad las soluciones lógicas a sus objetivos.

Lógica de Programación

- Con los elementos conceptuales de la lógica de programación es posible:

- Resolver ***cualquier problema*** o alcanzar cualquier objetivo que pueda ser ejecutado a través de computadoras.

- No solamente cualquier objetivo computacional, si no que también bajo ***cualquier lenguaje de programación.***

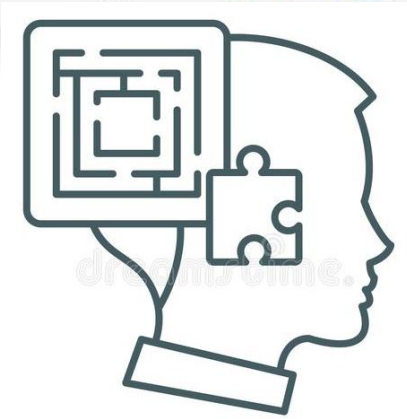
Lógica de Programación

- *"Buscar soluciones muy lógicas utilizando unos conceptos muy sencillos"*

- *"La lógica de programación es la unión de muchos (pero muchos) conceptos sencillos para el diseño de soluciones muy (pero muy) lógicas"*

Ph.D. Omar Iván Trejos Buriticá

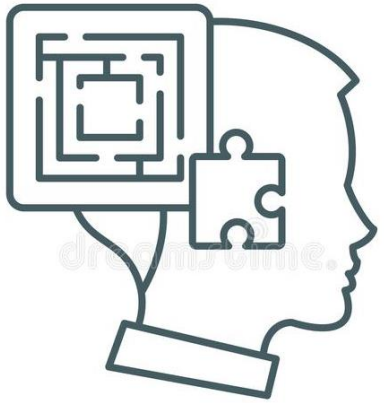
Lógica (Algunas Definiciones)



Su concepto es claro (obvio, tal vez), pero es difícil de definir:

- **Diccionario:** Lógica, es la rama del conocimiento que nos permite determinar que algo está aprobado por la razón como bien deducido o bien pensado.
- **Definición Matemática:** Ciencia que estudia la estructura, fundamentos y uso de las expresiones del conocimiento humano.
- **Definición Informal:** Serie coherente de ideas y razonamientos.

Lógica

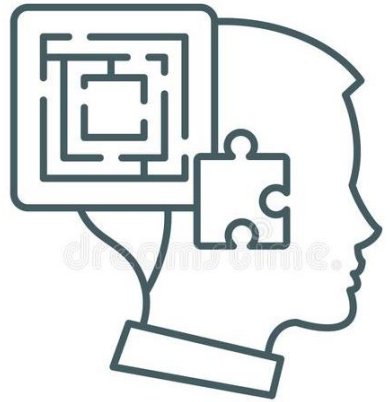


*"Forma más **OBVIA** y más **FÁCIL** de hacer algo."*

Ph.D. Omar Iván Trejos Buriticá

Metodología Para Solucionar un Problema

- *Problema* -> **Encontrar una solución**
- Existe un camino estructural para resolver cualquier problema (en términos generales)
- Dependiendo del problema, se detallan los pasos para alcanzar la solución del mismo



Metodología Para Solucionar un Problema

1. Objetivo

- Conocer muy bien cuál es el problema
- El problema entonces se tomará como el **Objetivo**
- Beneficios de identificar el **Objetivo**:

1. Tener claro el objetivo nos permite saber hacia dónde vamos.
2. Tener claro el objetivo nos permite saber hasta dónde debemos llegar (delimitar).

Metodología Para Solucionar un Problema

2. Algoritmo

- Conjunto de ***pasos secuenciales y ordenados*** que permiten lograr un ***objetivo***.
- ***Secuenciales*** significa que deben ser ejecutados uno después de otro y que sean pasos ***ordenados,*** quiere decir que deben llevar un orden cuasi-obligatorio (u obligatorio en la mayoría de los casos).

Ejemplo Algoritmo

- Desarrollar un algoritmo que nos permita adquirir el libro “El Coronel no tiene quien le escriba” de Gabriel García Márquez.

1. Objetivo

2. Algoritmo

Ejemplo Algoritmo

1. Objetivo

- Adquirir el libro “El Coronel no tiene quien le escriba” de Gabriel García Márquez.
- Mucha atención al objetivo! **Solamente es adquirirlo**, en ningún momento el objetivo es leerlo o resumirlo ni nada, solamente adquirirlo.

Ejemplo Algoritmo

2. Algoritmo

- Salimos del lugar en donde estemos y nos dirigimos hacia una librería. En caso de que ya estemos en una pues sencillamente solicitamos si tienen el libro, si lo tienen lo adquirimos y si no lo tienen vamos a otra librería en donde repetimos el proceso.

Generalización del Algoritmo

Algoritmo Adquisicion_Libro
Inicio

1. *Saber cuál es el libro que se quiere adquirir*
2. *Desplazarnos hacia una librería*
3. *Preguntar si tienen el libro que necesitamos*
4. *Si lo tienen*
   adquirirlo y Parar allí (dentro de este algoritmo)
   Si no lo tienen
   ir al paso 2

Fin

Observaciones Generales del Algoritmo

- Casi todas las líneas van numeradas, pero no todas.

- En la línea 1 se debe cumplir esa orden para poder continuar con el resto del algoritmo, porque se asume en algoritmo que no sólo se pasa por encima de las líneas sino que se realizan las tareas allí indicadas.

- Si realizamos todos los pasos que indica este algoritmo, podremos obtener el libro que sea porque la connotación de éste es absolutamente genérico sin restricciones, ya que en ningún momento se está diciendo que nos desplacemos hacia una librería que quede en la ciudad.

Observaciones Generales del Algoritmo

- Si luego de recorrer todas las librerías de todos los países de todo el mundo vimos que no pudimos conseguir el libro entonces podemos obtener dos conclusiones: Una es que el libro que buscábamos no lo tiene ninguna librería porque está agotado y la otra es que el libro es posible que nunca haya existido.

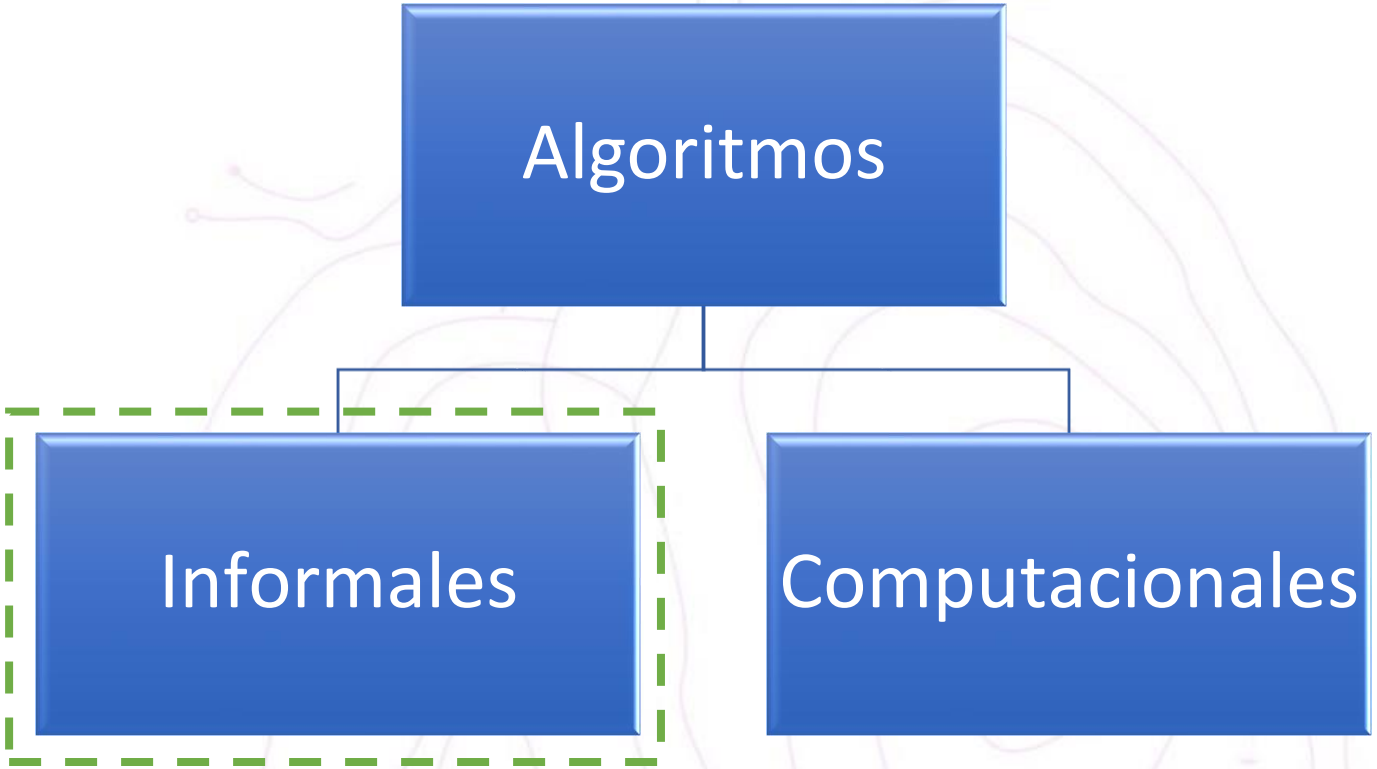
- Si probamos este ejemplo con el libro en mención (o sea El Coronel no tiene quien le escriba) tendremos un alto porcentaje de seguridad de que lo conseguimos a menos que esté agotado...

Puesta en Marcha del Algoritmo

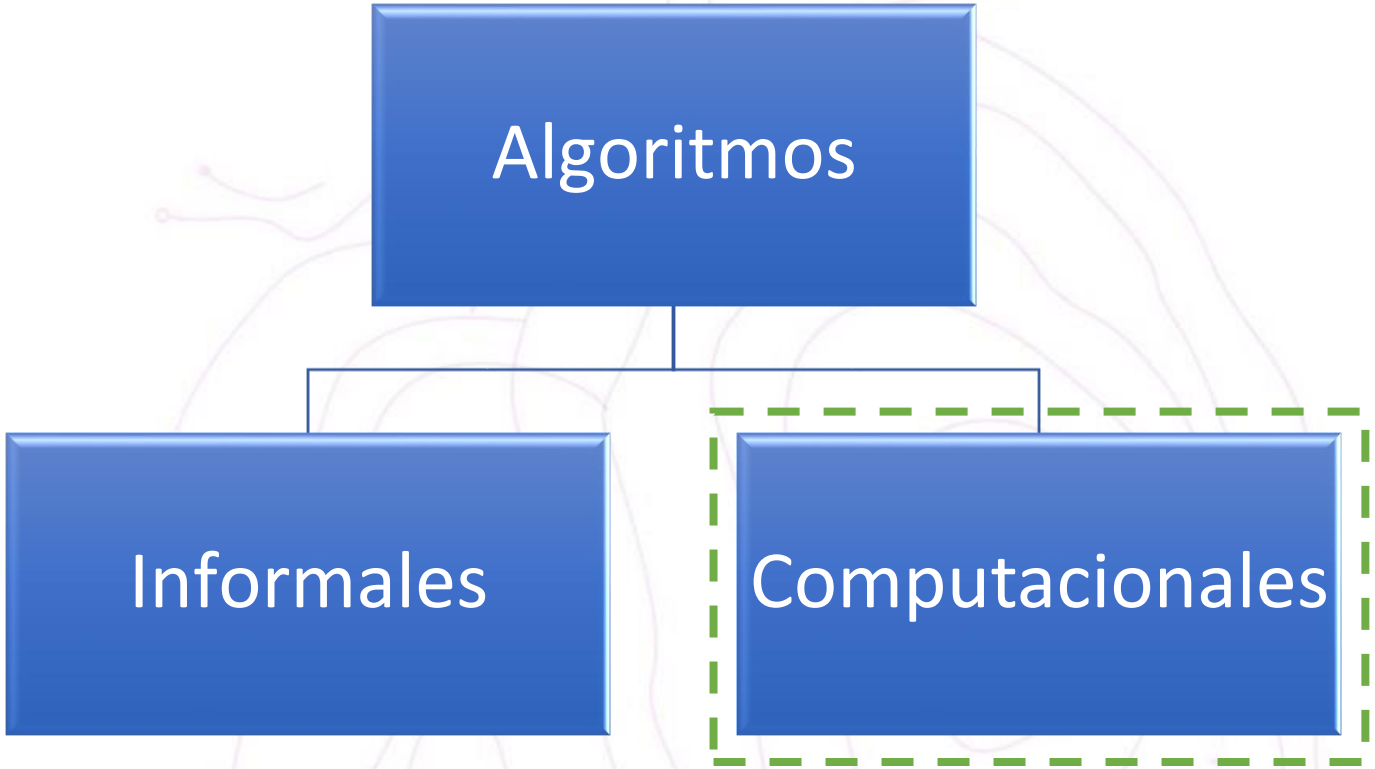
3. La Prueba

- Simular el funcionamiento del mismo.
- Busca verificar que los pasos alcancen el ***Objetivo***.
- Si ***no se alcanza el objetivo***, debe reformularse hasta que los pasos lleven a la meta delimitada.

Clasificación Algoritmos



Clasificación Algoritmos



Universidad Tecnológica de Pereira

Mision TIC 2022

El futuro digital es de todos

MinTIC

Algoritmos

Algoritmos Computacionales

Hechos QUE CONECTAN

Algoritmos Computacionales

- Algoritmos que deben ser preferiblemente implementados en un computador para aprovechar su *velocidad de procesamiento*.

- **Ejemplo:** algoritmo que genere los primeros 100 números primos.

- El computador lo puede hacer más rápido y sin errores si el algoritmo es construído correctamente.

Pasos para construir un Algoritmo Computacional

Posterior a la conceptualización (solución o algoritmo informal):

1. Transcripción
2. Digitación
3. Compilación
4. Ejecución o Puesta en Marcha
5. Verificación de Resultados

Pasos para construir un Algoritmo Computacional

1. Transcripción

- Traducción de un algoritmo con la ortografía (reglas sintácticas) de un Lenguaje de Programación.
- Busca verificar que los pasos alcancen el **Objetivo**.
- Un **programa** es un algoritmo computacional escrito bajo las reglas de un Lenguaje de Programación.

Pasos para construir un Algoritmo Computacional

2. Digitación

- Escribir en el computador el programa que ha sido escrito en el papel.
- Un programa no es más que un texto escrito bajo las reglas sintácticas de un lenguaje de programación.
- Se recomienda escribir el ***programa*** en papel para mejorar la implementación del algoritmo.

Pasos para construir un Algoritmo Computacional

3. Compilación

- Es el proceso a través del cual el computador revisa que el programa que hemos digitado se ajuste a las reglas sintácticas de un determinado Lenguaje de Programación
- El **Compilador** (programa) realiza este proceso.
- El Compilador busca **Errores de Sintaxis** y **Errores de Precaución**.

Pasos para construir un Algoritmo Computacional

4. Puesta en Marcha

- Cuando no hay **Errores de Sintaxis** y **Errores de Precaución**, se procede a “correr” el programa, es decir, que el computador lo ejecute (siga las instrucciones).

- En esta etapa, debería alcanzarse el objetivo trazado inicialmente al diseñar el algoritmo

Pasos para construir un Algoritmo Computacional

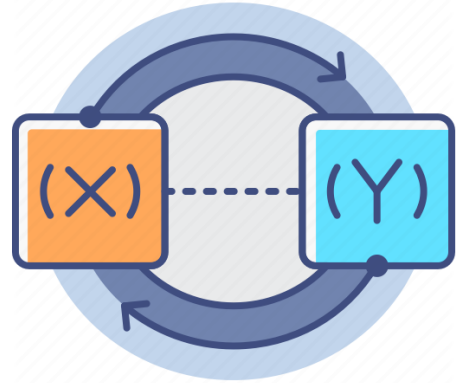
5. Verificación de Resultados

- Consecución del objetivo.
- Si no se consigue el objetivo fue porque:
  a) No teníamos claro el objetivo y fallamos en todo el proceso
  b) No realizamos bien la prueba de escritorio y nos la saltamos creyendo que el algoritmo estaba bien
  c) No conocíamos bien las reglas sintácticas del lenguaje utilizado

Variables, Algoritmo y Sistema

- **Algoritmo:** Serie de pasos lógicos para alcanzar un objetivo, la solución a un problema.

- Las **variables** representarán el punto del que se parte para alcanzar el **objetivo**, y los pasos del algoritmo son las acciones que modificarán el estado de estas.

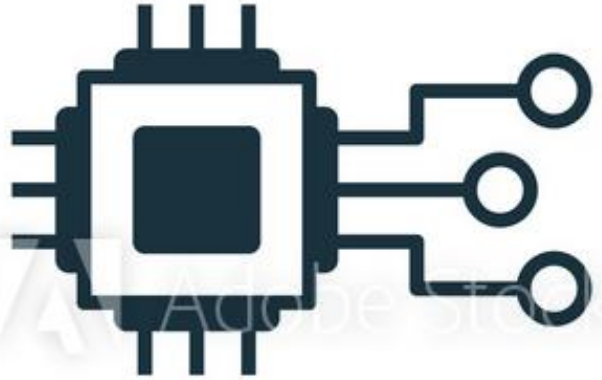


- El **estado final de las variables** contiene la **solución u objetivo** que queremos alcanzar.

Variables, Algoritmo y Sistema

- De una manera más amplia, un **algoritmo** puede verse entonces como un **sistema.**

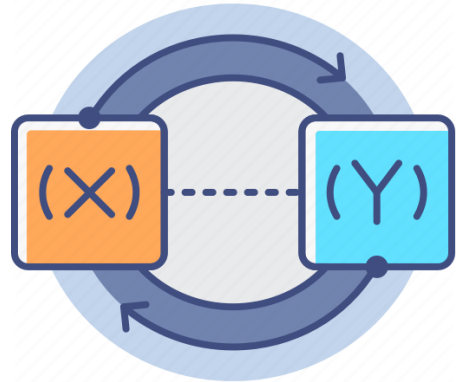
- **Sistema**: Información almacenada/representada en unas variables, son transformadas para ofrecer unas salidas (objetivo).



Variables

- Un **campo de memoria** que se reserva con un **nombre** al que se le puede cambiar su **contenido** cuantas veces sea necesario.
- Una **variable** tiene entonces un **nombre**, y almacena un **valor** de un **tipo de dato** específico.

NAME	VALUE	TYPE
number	123	int
sum	-456	int
pi	3.1416	double
average	-55.66	double



Asignando Valores a Variables

a = 8	Le indica al computador que guarde la constante 8 en la variable a
b = a	Le indica al computador que guarde en la variable b el contenido de la variable a que en la instrucción había sido "cargada" con 8, por lo tanto en la variable b queda el valor de 8 al igual que en la variable a
c = a + b	Le indica al computador que guarde en la variable c el resultado de sumar el contenido de la variable a con el contenido de la variable b. Como la variable a tenía el contenido 8 y la variable b también tenía el contenido 8 entonces el computador sumara 8+8 y ese 16 de resultado lo almacenará en la variable c

Ejemplo Instrucciones - Variables

Entero: A, B, C	Declara de tipo entero las variables A, B y C de manera que solo podrán almacenar datos enteros
A = 10	Almacena la constante 10 en la variable A
B = 15	Almacena la constante 15 en la variable B
C = 20	Almacena la constante 20 en la variable 20

Ejemplo Instrucciones - Variables

A = A + B	Almacena en la variable A el resultado de sumar el contenido de A mas el contenido de B o sea 10+15 que es igual a 25
B = B + 8	Almacena en la variable B el resultado de sumar el contenido de B con la constante 8 o sea 15+8 que es igual a 23
C = C + A	Almacena en la variable C el resultado de sumar el contenido de la variable C mas el contenido de la variable A o sea 20+25 que es igual a 45. Recuérdese que en esta línea se utiliza el último valor de almacenado en la variable A

Ejemplo Instrucciones - Variables

A = A + 5	Almacena en la variable C el resultado de sumar el contenido de la variable A mas la constante 5 es decir 25+5 que es igual a 30
B = B + 3	Almacena en la variable B el resultado de sumar el contenido de la variable B mas la constante 3 o sea 23+3 que es igual a 26
C = C + 2	Almacena en la variable C el resultado de sumar el contenido de la variable C mas la constante 2 o sea 45+2 que es igual a 47

Ejemplo Instrucciones - Variables

A = A – B	Almacena en la variable A el resultado de restarle al contenido de la variable A el contenido de la variable B o sea 30-26 que es igual a 4
B = A – B	Almacena en la variable B el resultado de restarle al contenido de la variable A el contenido de la variable B o sea 4-26 que es igual a -22

Ejemplo Instrucciones - Variables

C = A – B	Almacena en la variable C el resultado de restarle al contenido de la variable A el contenido de la variable B o sea 4 (-22) que por propiedades algebraicas es igual a 4+22 o sea 26

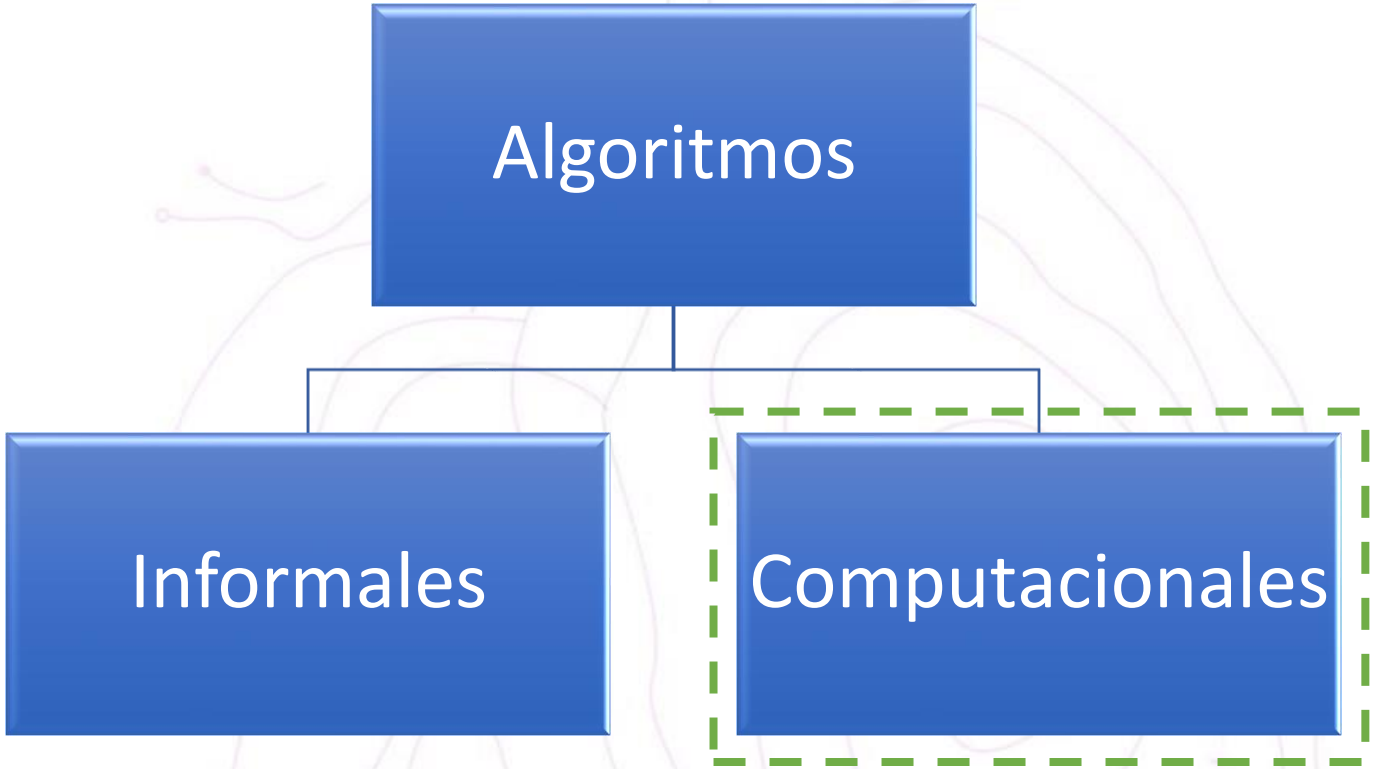
Los resultados finales en las tres variables son

Variable A	4
Variable B	-22
Variable C	26

Ejemplo Instrucciones - Variables

Entero: A, B, C	**A**	**B**	**C**
A = 10	10		
B = 15	10	15	
C = 20	~~10~~	15	20
A = A + B	25	~~15~~	20
B = B + 8	25	23	~~20~~
C = C + A	~~25~~	23	45
A = A + 5	30	~~23~~	45
B = B + 3	30	26	~~45~~
C = C + 2	~~30~~	26	47
A = A – B	4	~~26~~	47
B = A – B	4	-22	~~47~~
C = A – B	4	-22	26

Clasificación Algoritmos



Esquema

- *Esquema que nos permite representar de manera simplificada alguna idea, y que bajo condiciones normales, es constante.*

- Se acota o delimita la Lógica Algorítmica para estandarizar la representación de los algoritmos.

- Hablar un mismo idioma (o por lo menos similar).

Situaciones

1. ¿Con cuántos algoritmos las señoras de la casa pueden preparar los fríjoles?

Situaciones

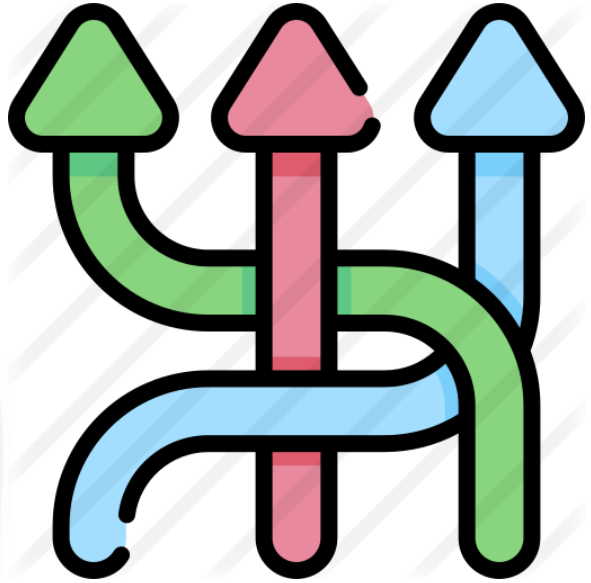
2. ¿Cuántas personas ve usted cerca que lleven puesta una camisa y un pantalón exactamente igual al suyo?

Variabilidad

- La concepción de un algoritmo varía desde su lógica hasta su aspecto técnico.

- ¿Qué pasaba si una persona concebía un algoritmo computacional en unas condiciones lógicas que prácticamente solo ella la entendiera?

- A nivel informal la variabilidad diversifica y genera nuevo conocimiento.



Objetivo Representación

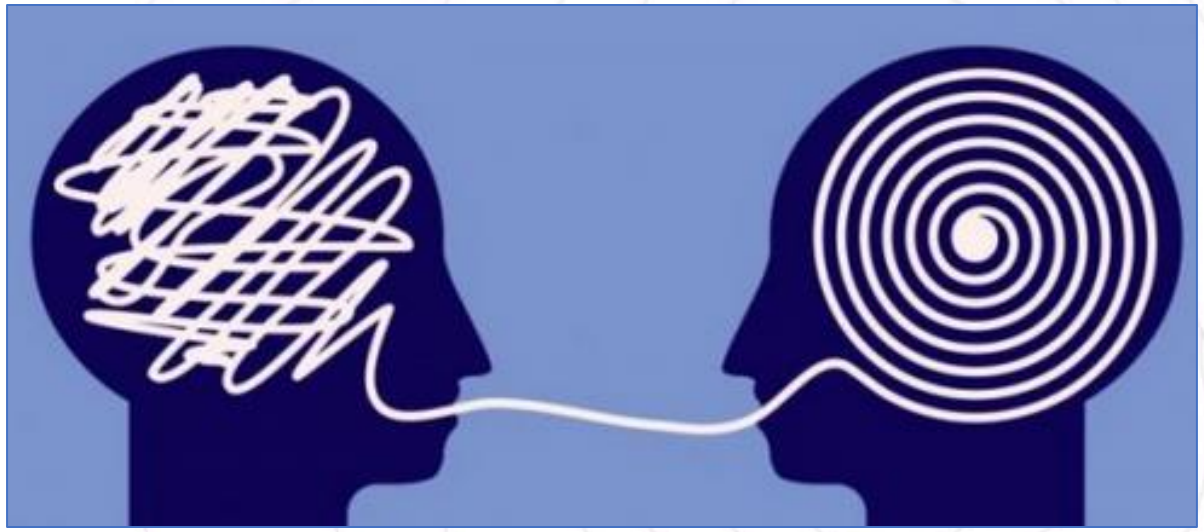
- **<u>Al estandarizar:</u>** Un programa desarrollado por una persona sea fácilmente entendible por cualquier otra.

- Podemos llegar a encontrarnos con programas tan confusos que solo llegarían a ser entendibles por su creador.

Objetivo Representación

- ***"Uniformar"*** la lógica para desarrollar algoritmos computacionales.

- Lógicas diferentes, expresadas en el mismo ***"idioma".***



Consideraciones Algorítmicas – Pensamiento Humano

- **Algoritmo Informal -> Algoritmo Computacional**

- Se identifican 3 estructuras básicas:

1) Secuencia
2) Decisión
3) Ciclos

Consideraciones Algorítmicas – Pensamiento Humano

1. Secuencia

Consideraciones Algorítmicas – Pensamiento Humano

1. Secuencia

- Permanentemente estamos inmersos en esta estructura.
- Primero **planeamos cada secuencia de acciones** (consciente o inconscientemente) **antes de ejecutarlas**.
- Cada una de las cosas que hacemos diariamente son **secuencias de acciones** que hemos planeado para poder alcanzar **objetivos**.

Consideraciones Algorítmicas – Pensamiento Humano

2. Decisión

Consideraciones Algorítmicas – Pensamiento Humano

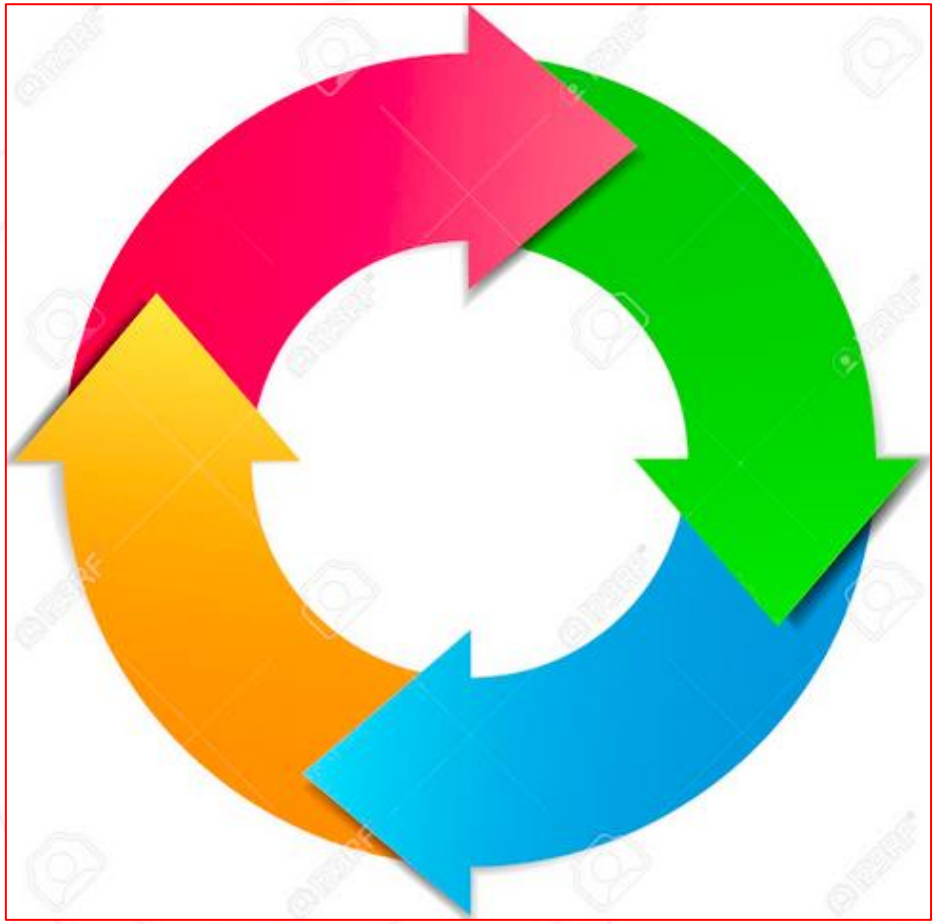
2. Decisión

- Posibles variantes en las secuencias, planes diferentes (pueden implicar ventajas y desventajas).

- La decisión se da siempre que usted tenga que **escoger de entre, por lo menos, dos caminos lógicos.**

Consideraciones Algorítmicas – Pensamiento Humano

3. Ciclos



Consideraciones Algorítmicas – Pensamiento Humano

3. Ciclos

- Se requiere realizar las secuencias más de una vez.
- Se requiere tomar decisiones más de una vez.
- Estructura que nos permite **repetir una o varias acciones una cantidad definida de veces.**

Consideraciones Algorítmicas – Pensamiento Humano

1. Secuencia Acciones

2. Decisión

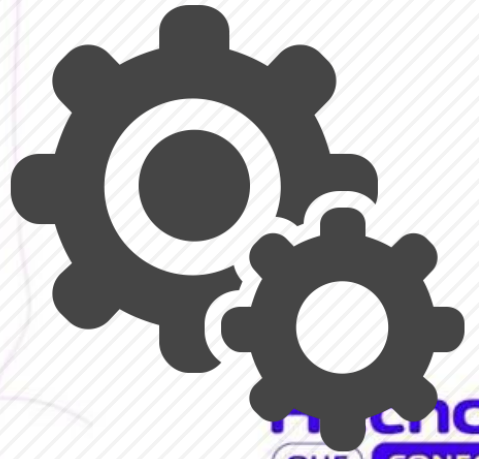
3. Ciclos

- Tomar una decisión depende de una **determinada condición** y que **repetir un conjunto de acciones** depende de que se **cumpla o se deje de cumplir igualmente una condición.**

Estructuras Básicas: Expresión Técnica

- Patrones técnicos que permitan ***describir las ideas lógicas de una manera uniforme.***

- Se han desarrollado unos esquemas que nos van a permitir escribir las estructuras mencionadas anteriormente.



Secuencias de Órdenes

- Para escribir una secuencia de órdenes o acciones todo lo que tiene que hacer es colocar una nueva orden o una nueva acción después de la última que haya colocado.

- La secuencialidad y la ordinalidad expresa el orden lógico en el que se deben ejecutar las acciones.

Secuencias de Órdenes
Ejemplo 1

- Desarrollar un algoritmo que nos permita asomarnos a la ventana, pero vamos a asumir que la ventana a donde nos queremos asomar ya está abierta y que no estamos muy distantes de la ventana.

Secuencias de Órdenes
Ejemplo 1

Algoritmo para asomarnos a la ventana
Inicio
Ubicar la ventana por la que nos queremos asomar
Levantarnos del lugar en donde estemos sentados
Avanzar hacia la ventana
Llegar hasta tener la ventana muy muy cerquita
Asomarnos por la ventana
Fin

Secuencias de Órdenes
Ejemplo 2

- Realizar un algoritmo para colocarnos una camisa (asumimos que la camisa está en nuestro ropero doblada y abrochada).

Secuencias de Órdenes
Ejemplo 2

Algoritmo para colocarnos una camisa
Inicio
Dirigirnos a nuestro ropero
Abrir el ropero
Tomar una camisa
Desabrocharla
Abrir la camisa
Meter un brazo por una de sus mangas
Meter el otro brazo por la otra de sus mangas
Ajustar la camisa al tronco
Abotonarla (botón a botón)
Fin

Secuencias de Órdenes Observaciones

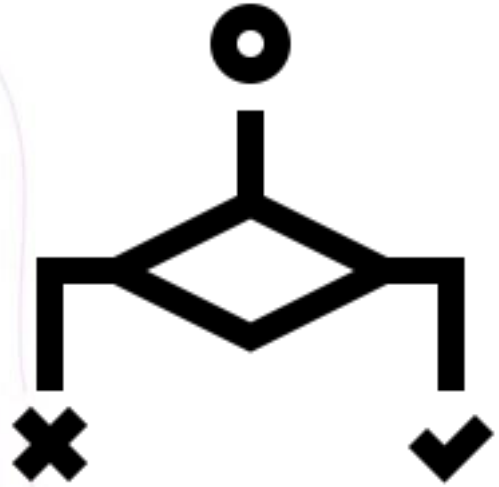
- En cuestión de algoritmos el ***orden de los factores sí altera el resultado.***

- La secuencia parece trivial, sin embargo, los ***errores en secuencia*** suelen ser ***difíciles de detectar*** al depurar un algoritmo codificado en un lenguaje de programación.

Las Decisiones

- Siempre que tengamos que utilizar la estructura de ***Decisiones*** vamos a depender de una ***condición***.

- La ***condición*** es la que nos permite que podamos decidir cuál es el ***camino lógico correcto a tomar***.



Las Decisiones
Ejemplo 1

- Mismo algoritmo para asomarnos a una ventana, pero esta vez no le vamos a colocar las condiciones de que estamos cerca de la ventana y de que ésta está abierta.

- Para ello vamos a incorporar una ***línea de decisión*** que nos permitan tener un algoritmo más genérico y que nos permita lograr mejor el objetivo.

Las Decisiones
Ejemplo 1

```
Algoritmo para asomarnos a la ventana
Inicio
        Ubicar la ventana por la que nos queremos asomar
        Si estamos sentados
                Levantarnos del lugar en donde estemos sentados
                Orientarnos hacia la ventana
        Sino
                Orientarnos hacia la ventana
        Avanzar hacia la ventana
        Llegar hasta tener la ventana muy muy cerquita
        Si esta cerrada
                Abrirla
        Asomarnos por la ventana
Fin
```

Las Decisiones
Ejemplo 1 - Observaciones

- El algoritmo ahora tiene unas condiciones que le permiten ser una secuencia de acciones más racional.

- El algoritmo se convierte en algo más depurado y mucho más aproximado a la realidad.

Las Decisiones
Ejemplo 1 - Observaciones

- Las palabras **Si** que aparecen son exclusivamente condicionales y no afirmativas como pudiera pensarse en algunos casos.

- Después de cada **Si** condicional va una **condición** que es la que permite que se haga una cosa u otra. La **condición** regula las acciones que vienen después y que dependen del **Si** condicional inicial.

Las Decisiones
Ejemplo 1 - Observaciones

- Puede usted notar que una decisión completa involucra
  - Una pregunta que evalúa una condición
  - Un conjunto de acciones a realizar en caso de que la condición sea verdadera
  - Un conjunto de acciones a realizar en caso de que la condición sea falsa

- No siempre que exista un condicional Si debe existir un Sino asociado a él. Siempre que exista un Sino es porque está asociado a un Si condicional determinado.

Las Decisiones
Ejemplo 2

```
Algoritmo para colocarnos una camisa
Inicio
        Dirigirnos a nuestro ropero
        Si esta cerrado
                Abrirlo
        Tomar una camisa
        Si está abrochada
                Desabrocharla
                Abrir la camisa
        Si está doblada
                Desdoblarla
        Meter un brazo por una de sus mangas
        Meter el otro brazo por la otra de sus mangas
        Ajustar la camisa al tronco
```

```
        Si es una camisa de botones
                Abotonarla (botón a botón)
                Ajustarla al cuerpo
        Sino
                ajustarla de manera que quede bien puesta
Fin
```

Los Ciclos
Ejemplo 1

- Usted es un supervisor de una fábrica y que cada media hora, a lo largo de todo el día, debe estar vigilando determinada acción a través de una ventana.

- El algoritmo para cumplir su objetivo que es el de Vigilar (como supervisor de la fábrica) parte de una unidad muy sencilla y es Asomarse por una ventana.

Los Ciclos
Ejemplo 1

- Usted tendrá que asomarse por una ventana mientras no termine el día cada media hora y durante el tiempo que usted no esté asomado lo que tiene que hacer es seguir en su puesto de trabajo.

- De esta forma, y partiendo de lo que ya tenemos, usted podrá estructurar un algoritmo más o menos de la siguiente manera.

Los Ciclos
Ejemplo 1

```
Algoritmo para Vigilar desde una ventana
Inicio
        Llegar puntual a la hora de inicio de la jornada laboral
        Ubicarnos en nuestro escritorio
        Mientras no sea el fin del día
                Ubicar la ventana por la que nos queremos asomar
                Si estamos sentados
                        Levantarnos del lugar en donde estemos sentados
                        Orientarnos hacia la ventana
                Sino
                        Orientarnos hacia la ventana
                Avanzar hacia la ventana
                Llegar hasta tener la ventana muy muy cerquita
                Si esta cerrada
                        Abrirla
                Asomarnos por la ventana
                Regresar a nuestro escritorio
                Mientras no haya pasado Media Hora
                        Permanecer en nuestro escritorio
                Fin_Mientras
        Fin_Mientras
Fin
```

Los Ciclos Ejemplo 1 Observaciones

- La palabra **Mientras** establece en relación con una **condición** el inicio de un conjunto de acciones que se repiten precisamente **Mientras esa condición lo permita**.

- Todo Mientras (por efectos de clarificación del algoritmo) debe tener un finalizador que indique hasta donde llega el bloque de acciones que debemos repetir.

Los Ciclos Ejemplo 1 Observaciones

- La indentación o lo que corrientemente se conoce como el “Sangrado” del texto es decir el hecho de que algunas acciones estén mas adentro de la hoja que otras, representa que existen bloques de acciones que tienen una característica.

- Cada ciclo de acciones que se inicie con Mientras deberá tener un Fin_Mientras asociado y a su vez cada Fin_Mientras deberá finalizar uno y sólo un ciclo iniciado con Mientras.

Los Ciclos Ejemplo 1 Observaciones

Acerca del **Sangrado**:

- Las acciones contenidas entre el Inicio y el Fin indican que son las acciones que conforman el algoritmo en mención.
- Las acciones comprendidas entre Mientras no sea Fin del día y su correspondiente Fin_Mientras son el conjunto o bloque que se debe repetir (o iterar) precisamente mientras la condición sea Verdadera o sea Mientras no sea fin del día.
- La acción comprendida entre Mientras no haya pasado Media Hora y su correspondiente Fin_Mientras es la acción que se deberá realizar hasta cuando se complete media hora.

Universidad Tecnológica de Pereira

Mision TIC 2022

El futuro digital es de todos

MinTIC

Algoritmos

Representación: Diagramas de Flujo

Hechos QUE CONECTAN

Diagramas de Flujo

- ***Símbolos*** que nos permiten expresar de manera gráfica las ***secuencias, estructuras de decisión y ciclos*** para alcanzar un ***objetivo***.

- Al expresar visualmente la lógica del algoritmo, suele ser más fácil de comprender.

- Los símbolos para representar un algoritmo siguen un estándar más o menos flexible.

Diagramas de Flujo

- Existen **muchas herramientas** para construir este tipo de representaciones de nuestros algoritmos.
- Algunas generan código automáticamente.
- Se recomienda la siguiente herramienta gratuita: ***yED Graph Editor***
- Sección ***Flowchart*** dentro de esta aplicación

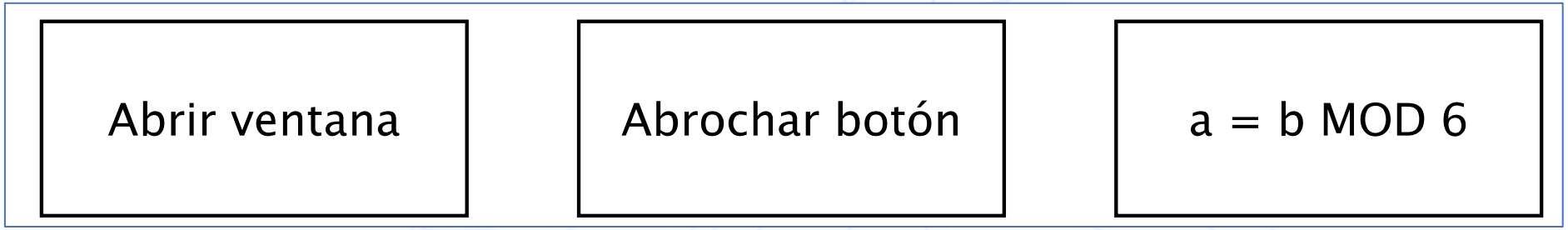
Diagramas de Flujo

- Enlace versión online ***yED Graph Editor***: https://www.yworks.com/yed-live/

- Sitio de descargas ***yED Graph Editor***: https://www.yworks.com/products/yed/download#download

- Seleccionar la descarga correspondiente al sistema operativo (Windows, MacOS o Linux)

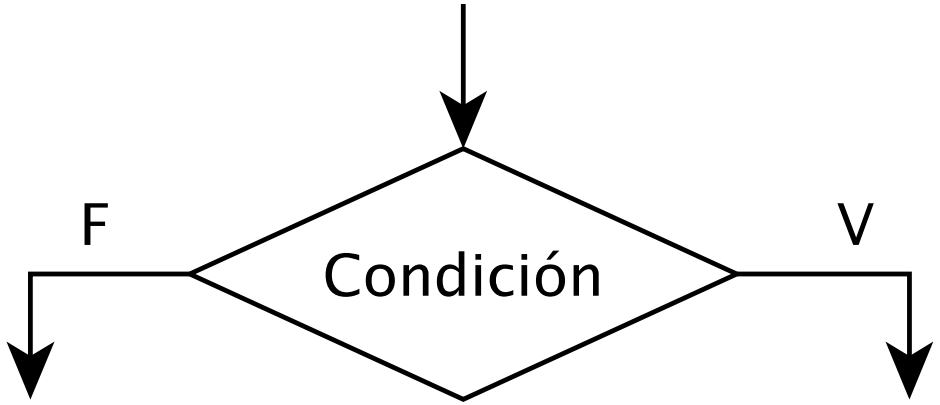
Diagramas de Flujo - Proceso



- Un rectángulo representa un proceso, que no es más que una ***acción*** o **una orden a ejecutarse de manera clara y concreta**.

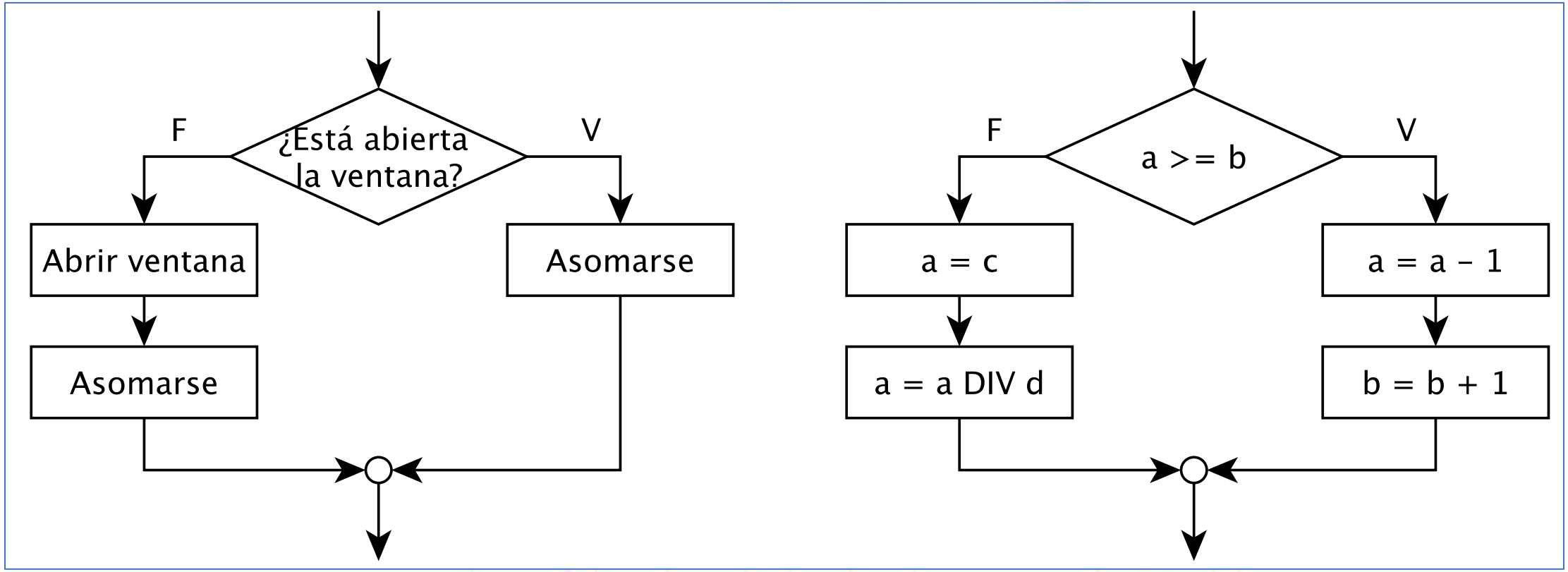
- Un ejemplo típico de proceso es: la asignación de un valor a una variable.

Diagramas de Fluio - Decisión

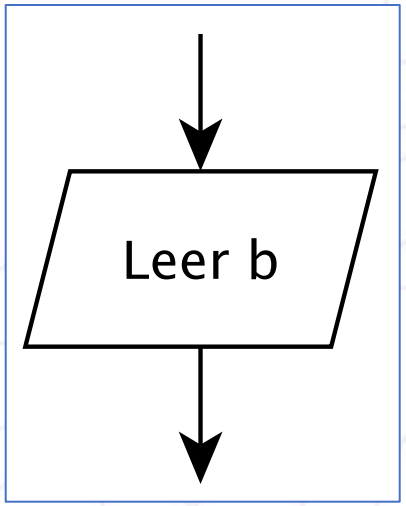


- Este símbolo nos permite representar una ***Decisión***.
- En su interior podemos escribir la ***condición***, de la cual depende la decisión, y por sus extremos derecho (o izquierdo) e inferior, se pueden colocar las salidas para los casos en que la condición sea ***Falsa*** o sea ***Verdadera***.

Diagramas de Flujo - Decisión Ejemplos

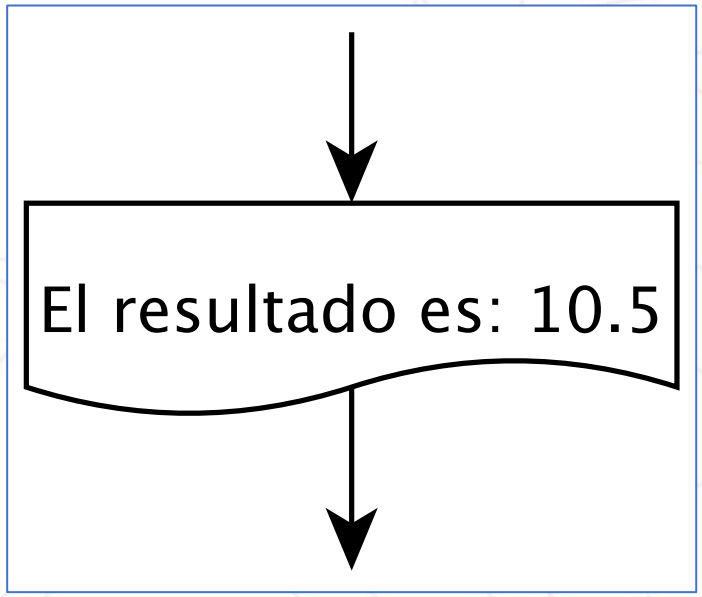


Diagramas de Flujo - Entradas o Salidas



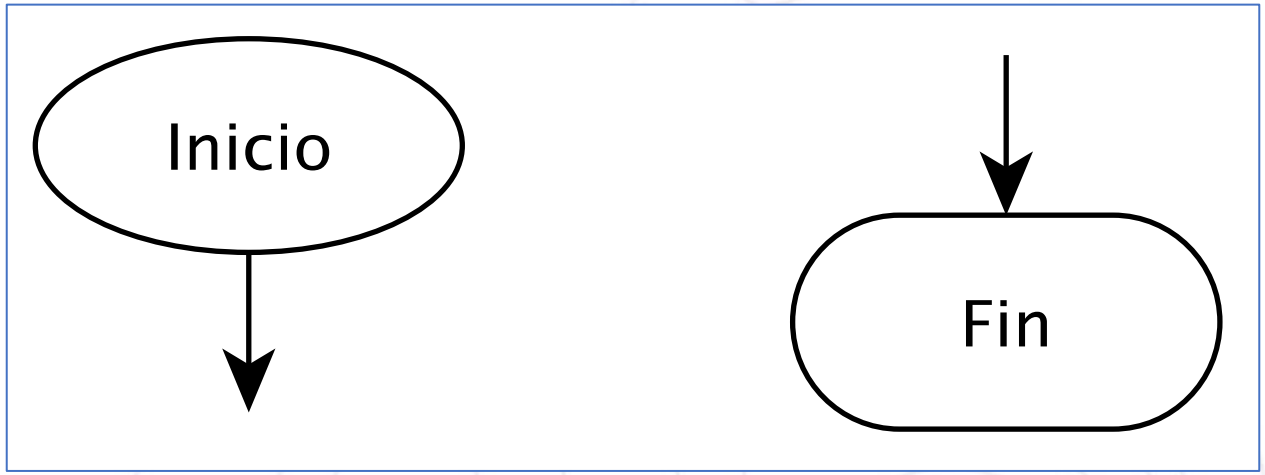
- Este símbolo nos permite expresar un proceso de ***entrada o salida***, teniendo en cuenta que una entrada en un algoritmo se concibe como el proceso a través del cual se recibe información y una salida es el proceso a través del cual se entrega información.

Diagramas de Flujo - Escritura o Salida



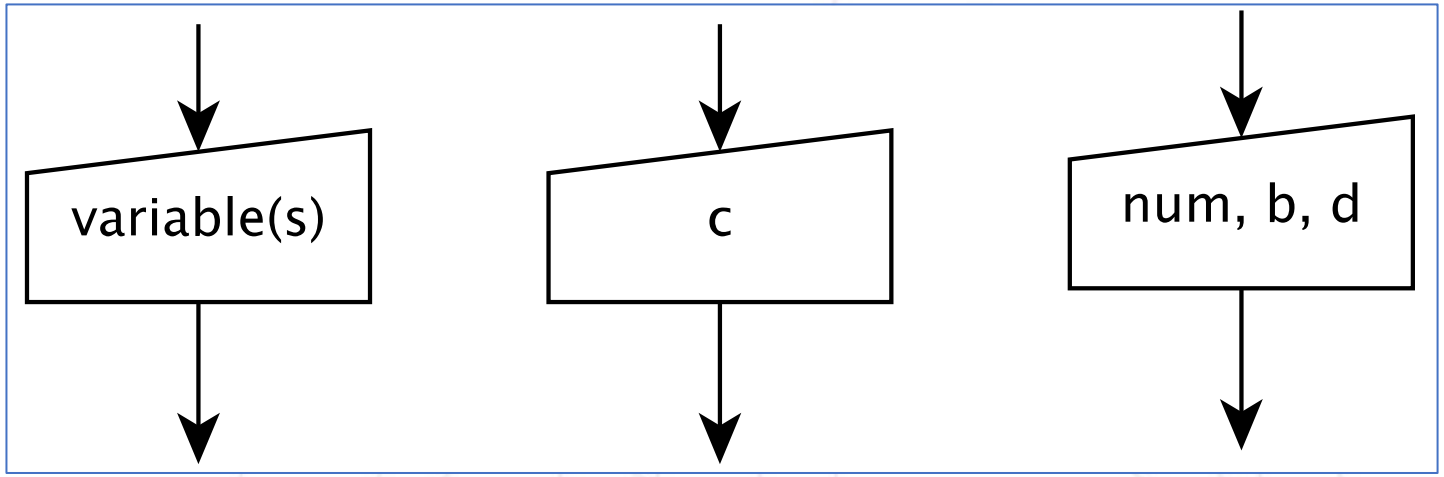
- Este símbolo permite representar la escritura en un dispositivo o periférico de un resultado, o lo que técnicamente se conoce como una ***salida (output)***.

Diagramas de Flujo - Inicio o Finalización



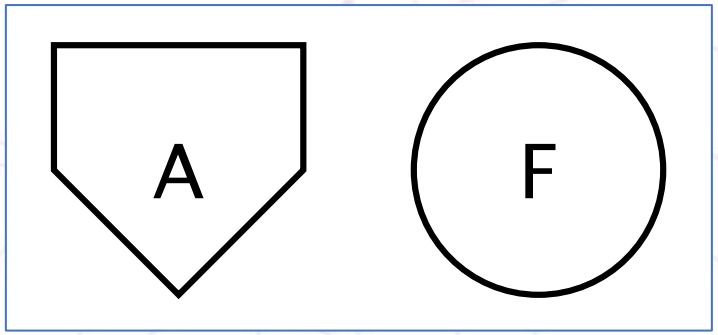
- Estos símbolos representan el ***Inicio*** ó el ***Fin*** de un Algoritmo (solución).
- Todo lo que se tiene que hacer es escribir la palabra Inicio o Fin y ubicarlo apropiadamente dentro del Diagrama de Flujo.

Diagramas de Flujo - Inicio o Finalización



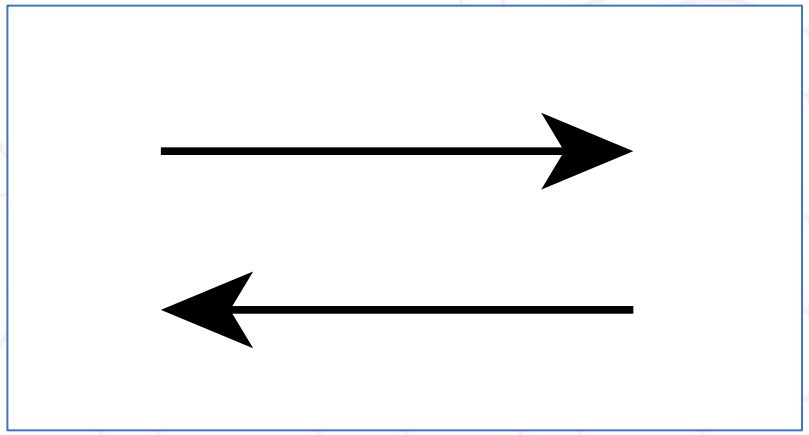
- Estos símbolos representan una ***entrada de datos utilizando el teclado del computador.***
- Todo lo que tenemos que escribir en su interior es el nombre de la variable (o las variables) en donde queremos que se almacene el dato que entra por el teclado.

Diagramas de Flujo - Conectores Diagramas



- Nos permiten representar la ***continuación*** de un Diagrama de Flujo cuando éste es tan largo que no cabe en una sola hoja.

Diagramas de Flujo - Flechas (Flujos)



- Las flechas son los símbolos que nos van a permitir representar la forma de conexión entre los demás símbolos determinando igualmente el ***Flujo de ejecución o realización de acciones***.

Ejemplo Previo

- Un algoritmo que nos permitiera Vigilar una empresa desde una ventana asomándonos cada media hora por ella.

Ejemplo Previo

```
1  //Nombre/Descripción del Algoritmo
2  Algoritmo para Vigilar desde una ventana
3  //Pseudocódigo del Algoritmo
4  Inicio
5      Llegar puntual a la hora de inicio de la jornada laboral
6      Ubicarnos en nuestro escritorio
7      Mientras no sea el fin del día
8          Ubicar la ventana por la que nos queremos asomar
9          Si estamos sentados
10             Levantarnos del lugar en donde estemos sentados
11             Orientarnos hacia la ventana
12         Sino
13             Orientarnos hacia la ventana
14         Fin_Si//Cierre condicional (Línea 9)
15         Avanzar hacia la ventana
16         Llegar hasta tener la ventana muy muy cerquita
17         Si esta cerrada
18             Abrirla
19         Fin_Si//Cierre condicional (Línea 17)
20         Asomarnos por la ventana
21         Regresar a nuestro escritorio
22         Mientras no haya pasado Media Hora
23             Permanecer en nuestro escritorio
24         Fin_Mientras//Cierre del Ciclo Mientras, correspondiente al paso de media hora (Línea 22)
25     Fin_Mientras//Cierre del Ciclo Mientras, correspondiente al transcurso del día (Línea 7)
26 Fin//Cierre del Algoritmo (Línea 4)
```

Diagrama de Flujo – Vigilar Ventana

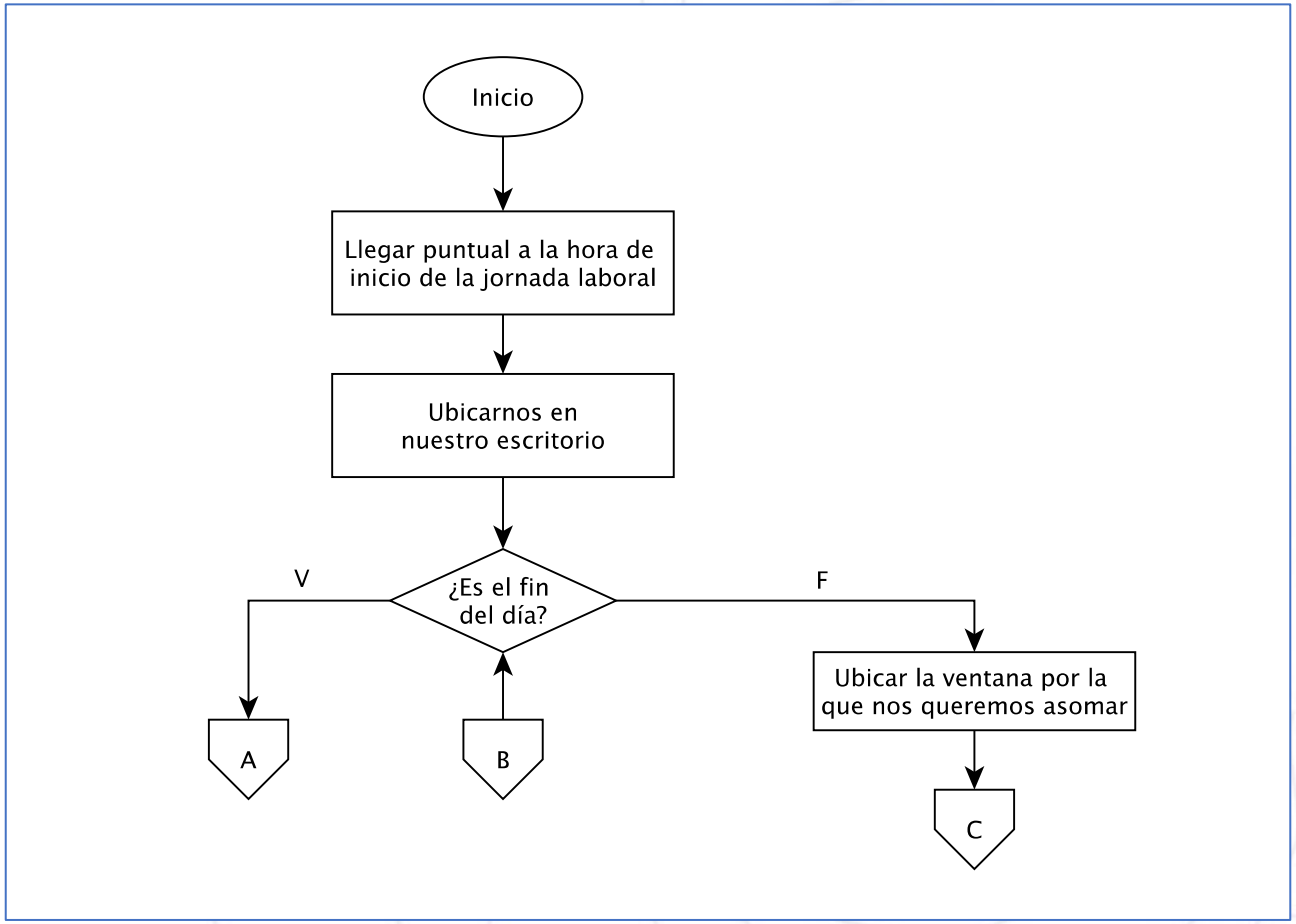


Diagrama de Flujo – Vigilar Ventana

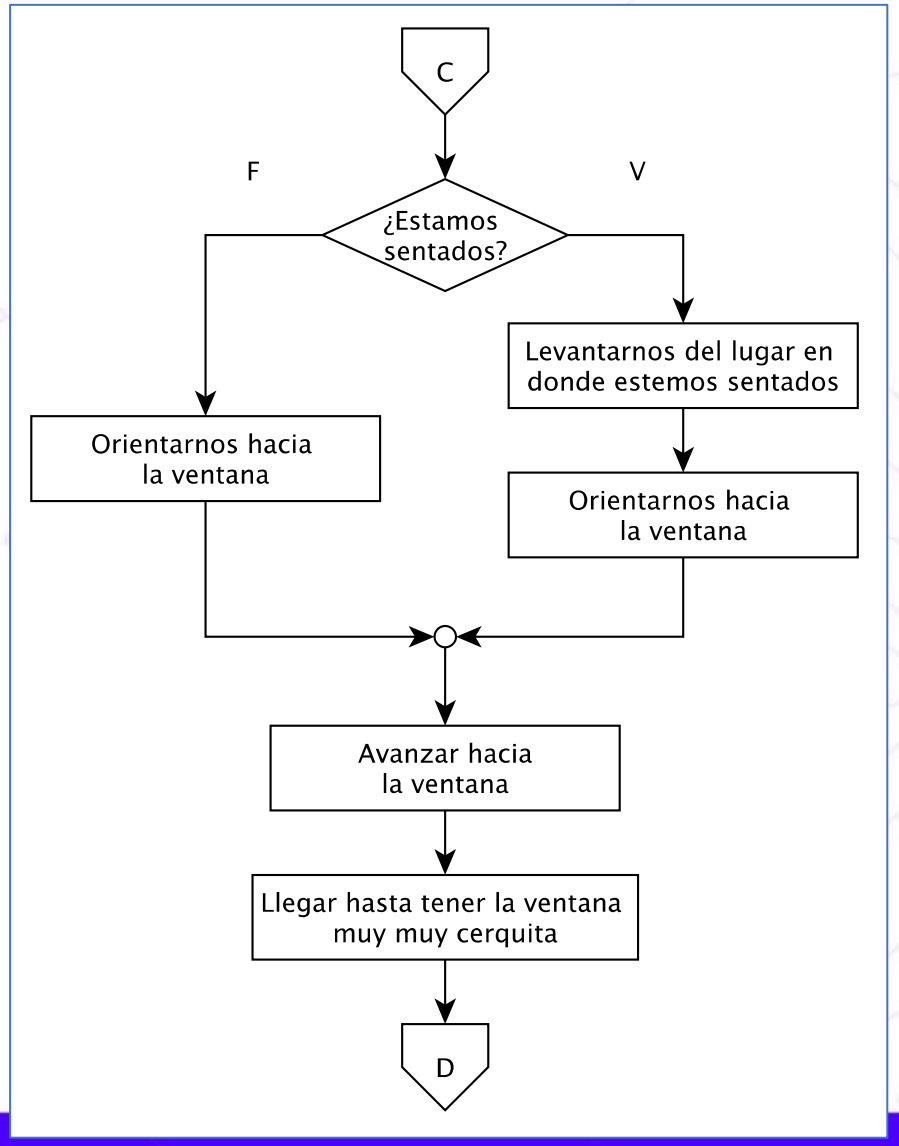


Diagrama de Flujo – Vigilar Ventana

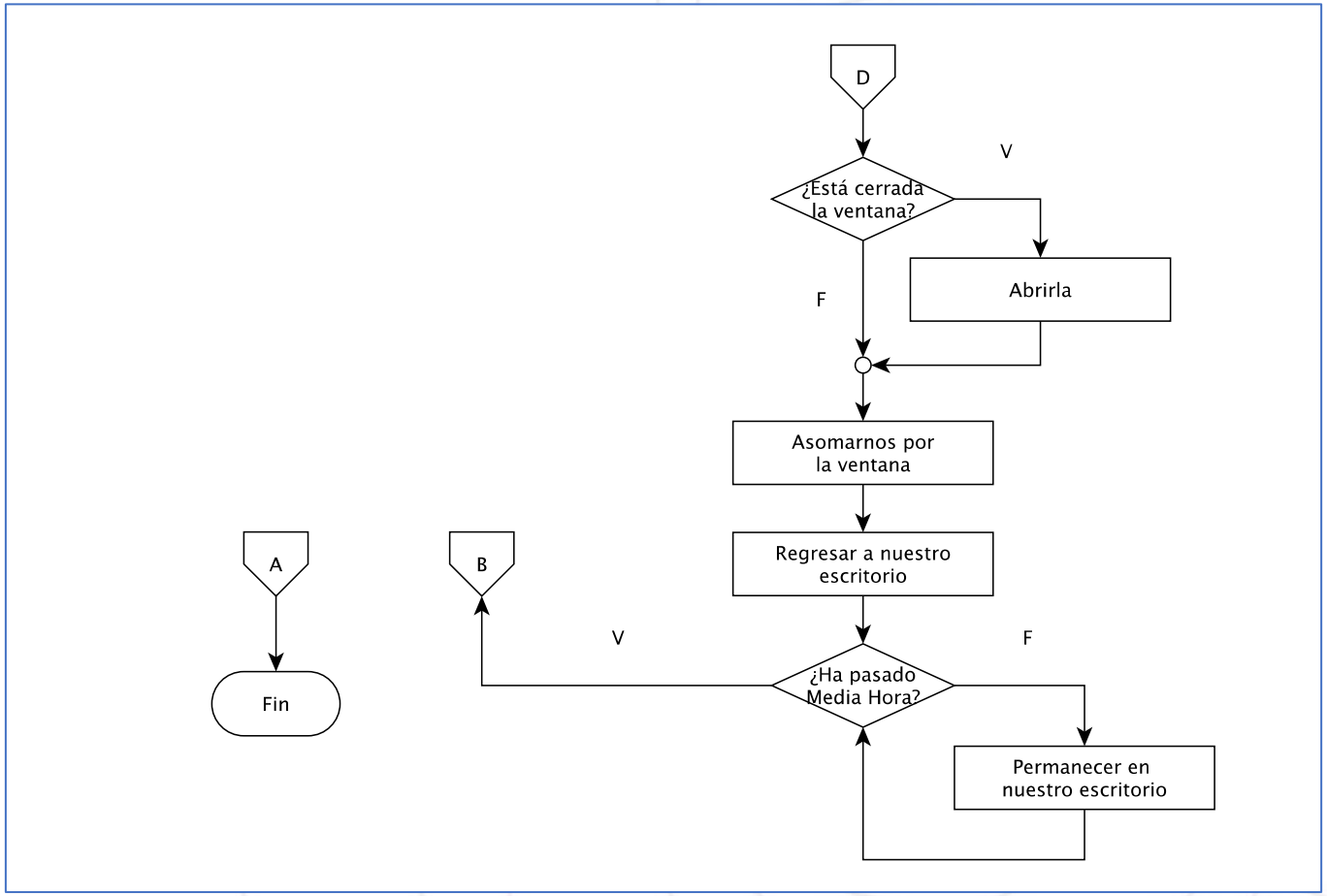


Diagrama de Flujo – Vigilar Ventana

- Visualizar el diagrama de flujo completo (se sugiere montarlo en yED).
- Observar el flujo de eventos e interconexiones.
- Observar jerarquía de las diferentes estructuras: ***lógica, decisiva y cíclica.***

Ejercicio: Proponer el Diagrama de Flujo

```
Algoritmo para Inspeccionar las camisas en un almacén de ropa
Inicio
        Llegar puntuales al inicio de la jornada laboral
        Mientras no sea fin de la jornada laboral
                Dirigirnos a un ropero
                Si esta cerrado
                        Abrirlo
                Tomar una camisa
                Si está abrochada
                        Desabrocharla
                        Abrir la camisa
                Si está doblada
                        Desdoblarla
                Meter un brazo por una de sus mangas
                Meter el otro brazo por la otra de sus mangas
                Ajustar la camisa al tronco
                Si es una camisa de botones
                        Abotonarla (botón a botón)
                        Ajustarla al cuerpo
                Sino
                        Ajustarla de manera que quede bien puesta
                Emitir el concepto de calidad sobre la camisa
        Fin_Mientras
Fin
```

Universidad Tecnológica de Pereira

Mision TIC 2022

El futuro digital es de todos

MinTIC

Algoritmos

Representación: Pseudocódigo

Hechos QUE CONECTAN

Pseudocódigo

- Sencillamente es la ***representación textual de un algoritmo*** de manera que dicho texto se encuentre enmarcado en ***algunas normas técnicas*** que faciliten su posterior ***transcripción a un Lenguaje de Programación.***

```
Algoritmo Serie
    Definir x como entero
    para x = 1 hasta 100
        Escribir x
    FinPara
FinAlgoritmo
```

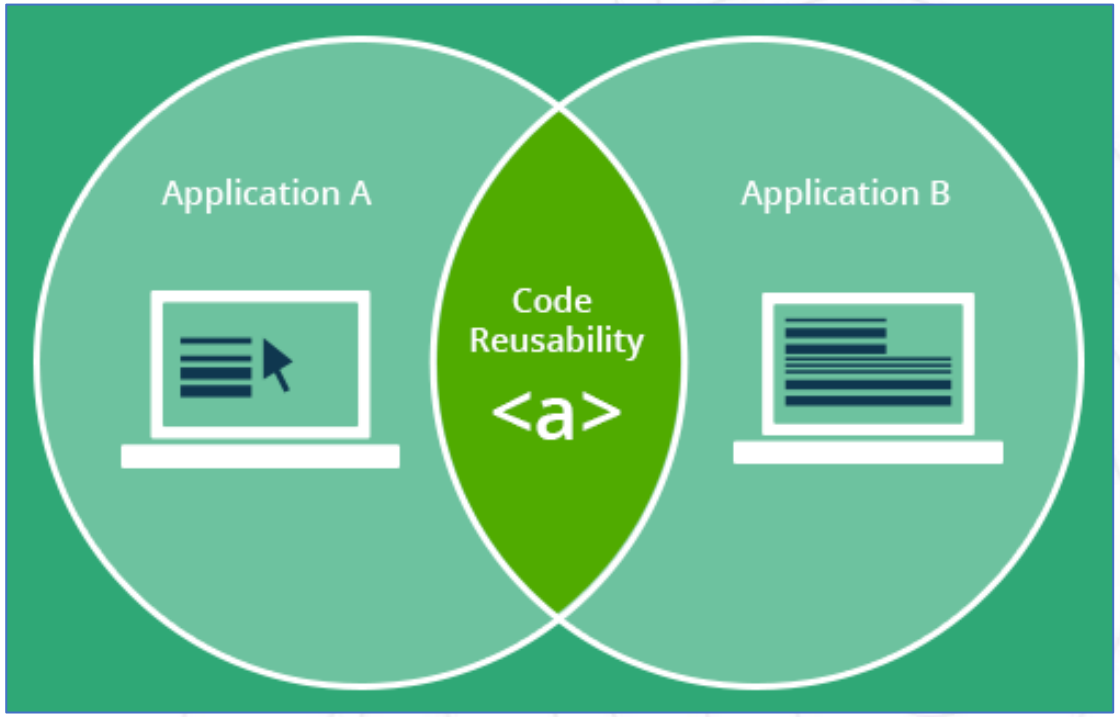
Normas Pseudocódigo

- Lineamientos que siendo opcionales, facilitan la posterior ***transcripción del algoritmo a un Lenguaje de Programación*** cuando son cumplidos a cabalidad.

- Las normas se requieren para eliminar ambigüedades o informalidad.

- El objetivo de las técnicas de representación es facilitar la posterior transcripción de los algoritmos.

Pseudocódigo Norma 1
Nombramiento

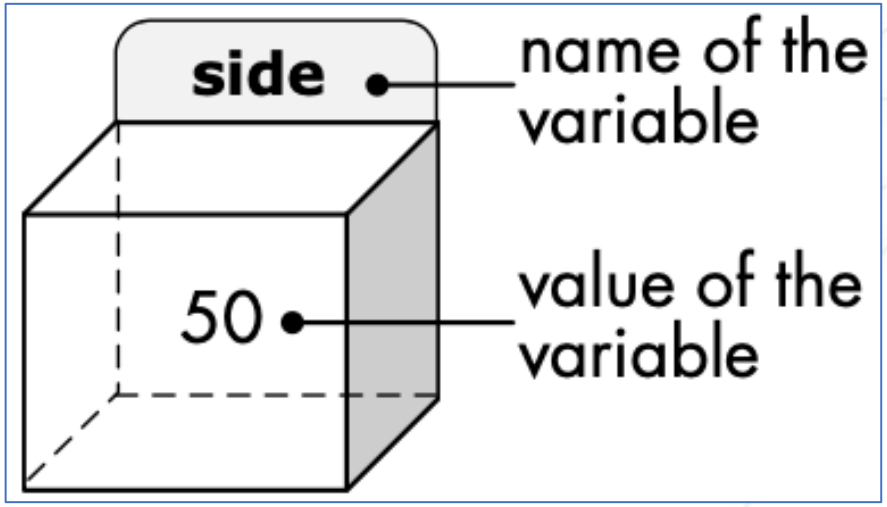


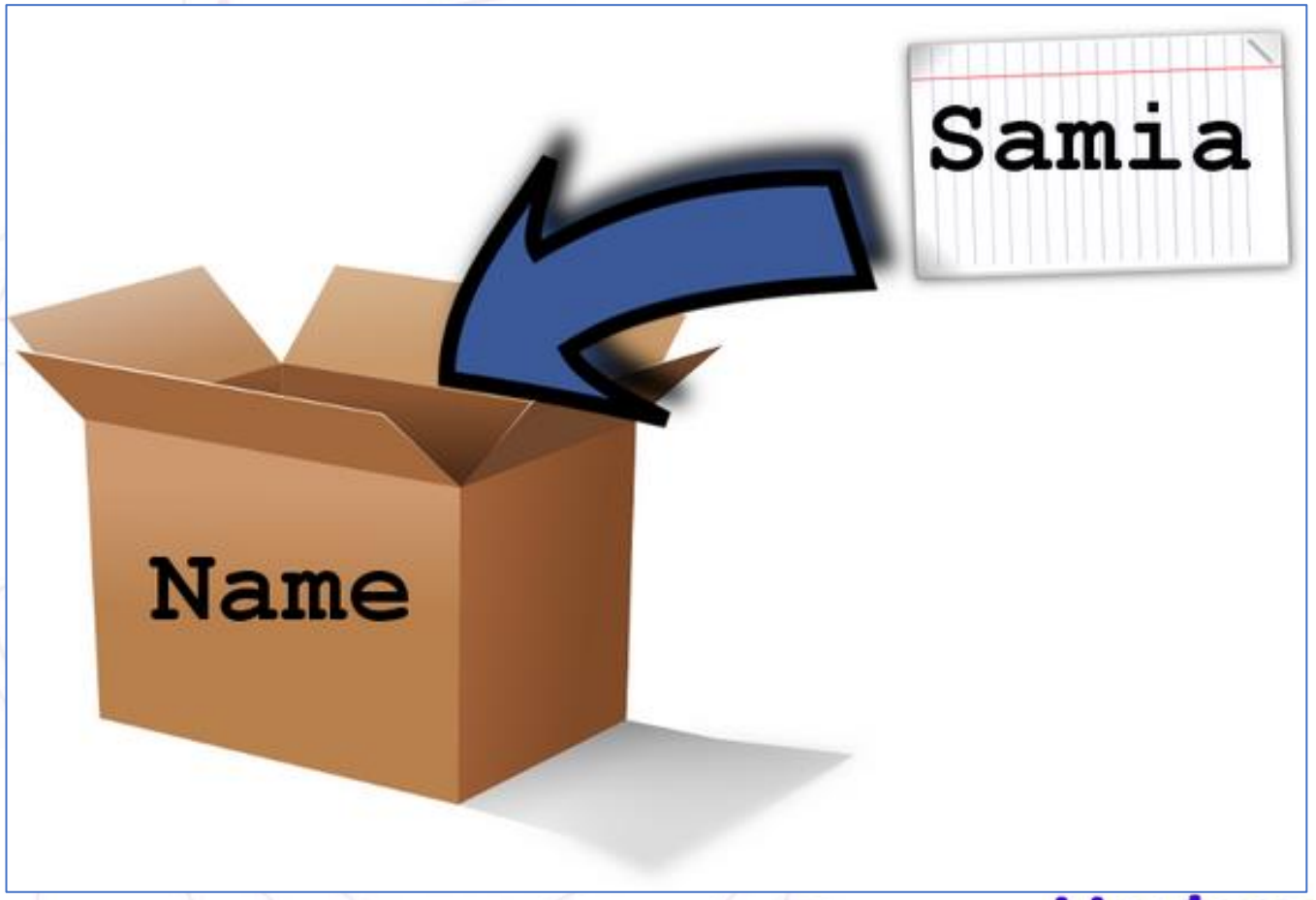
Pseudocódigo Norma 1
Nombramiento

- Establecer un nombre al algoritmo **corto** y que haga referencia aproximada a lo que contiene (mnemónico).

- La ***reutilización del código*** es un factor importante en el ejercicio profesional. Un nombre apropiado facilitará el aprovechamiento del esfuerzo realizado previamente, cuando se diseñaron algoritmos que solucionaron situaciones similares a las que tenemos en este momento, o por qué no, la misma situación.

Pseudocódigo Norma 2
Declaración Variables





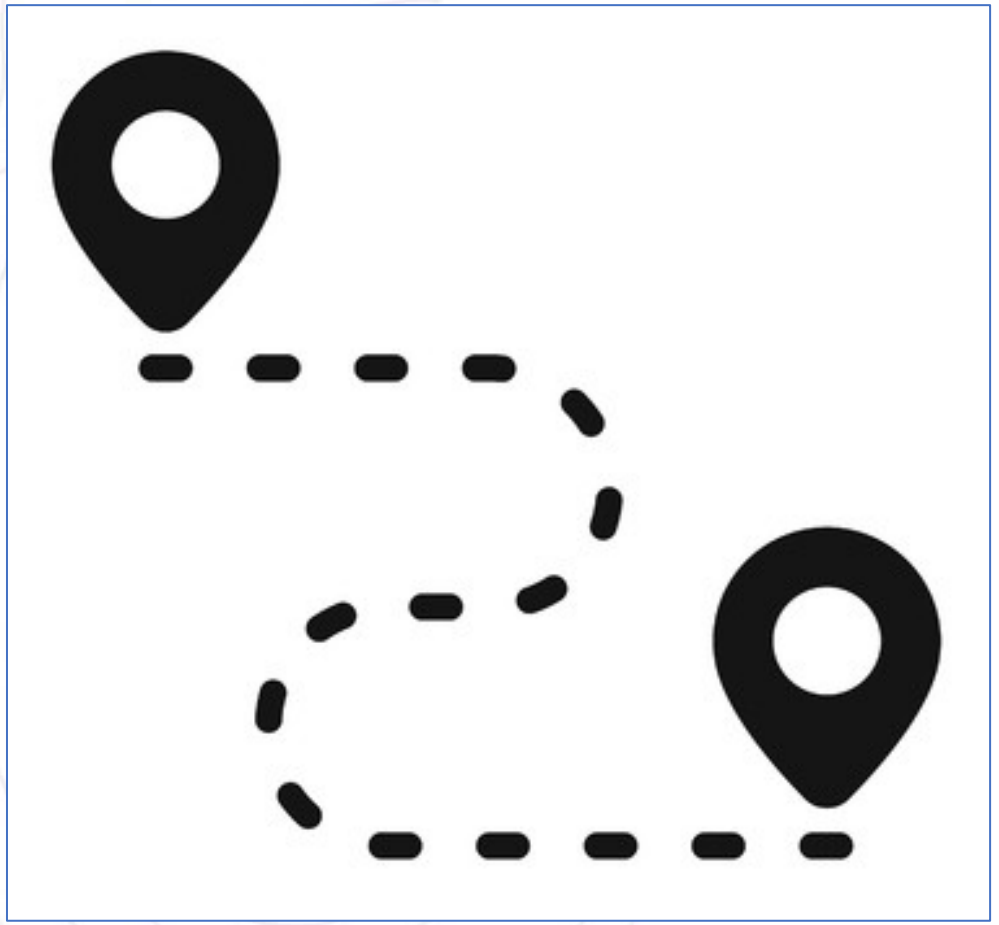
Pseudocódigo Norma 2
Declaración Variables

- Después de establecer un nombre con las características recomendadas, es recomendable ***especificar (declarar) todas las variables que se van a utilizar.***

- Declararlas significa escribir el tipo de dato que van a almacenar y el nombre que dichas variables van a llevar.

- Tipos de dato básicos: números enteros (int), números reales (float) y caracteres o texto (str).

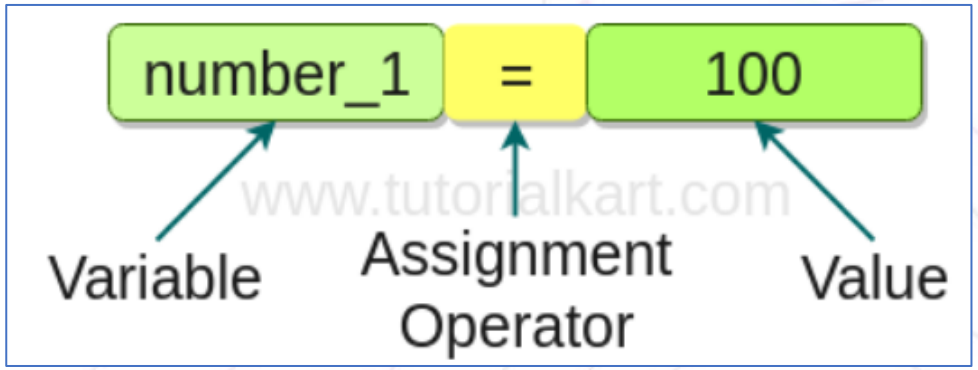
Pseudocódigo Norma 3
Inicio y Finalización

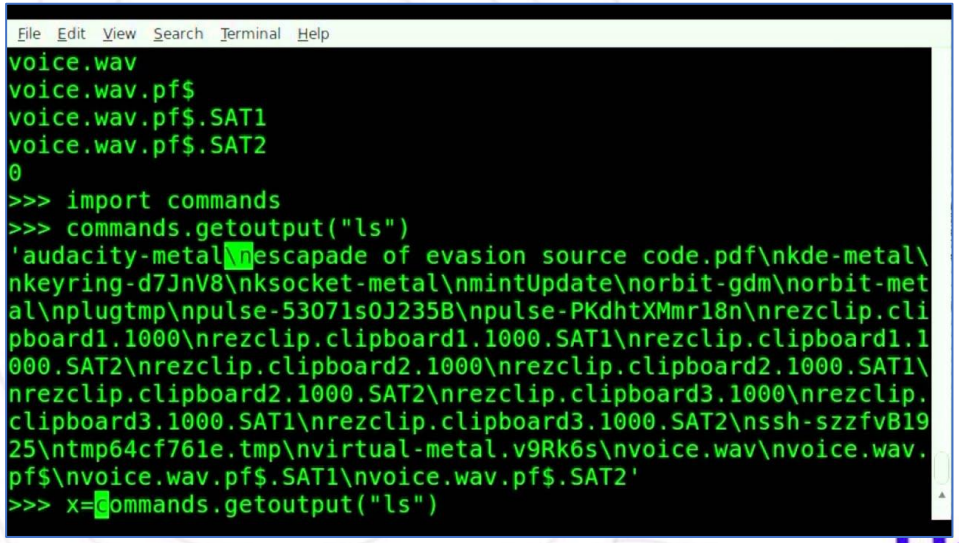


Pseudocódigo Norma 3
Inicio y Finalización

- Todo el cuerpo del algoritmo deberá ir *"encerrado"* entre las palabras ***Inicio y Fin*** indicando en donde comienza y en donde termina el pseudocódigo.
- De la misma forma, las estructuras de decisión, los ciclos y sus respectivas anidaciones, deben tener delimitados tanto sus inicializaciones como finalizaciones, además de la forma como se contienen entre sí cuando se presenta dicha anidación.
- También se refiere a delimitar y enmarcar algoritmos dentro de funciones.

Pseudocódigo Norma 4
Asignaciones y Mensajes





Pseudocódigo Norma 4
Asignaciones y Mensajes

- Para mostrar mensajes o salidas en pantalla, se utiliza la orden “Escriba” y a continuación se coloca entre comillas lo que se quiere mostrar en pantalla o informar al usuario durante el algoritmo.

- Por ejemplo:

  Escriba “Esta es una demostración”

Generará en pantalla el título: *Esta es una demostración*

Pseudocódigo Norma 4
Asignaciones y Mensajes

- Si se requiere que en pantalla se presente el contenido de una variable, se utiliza la orden “Escriba” y a continuación y sin comillas el nombre de la variable correspondiente. Esto se refiere al contenido de la variable.
- Por ejemplo:

```
N = 5
Escriba N
```

Mostrará en pantalla el valor: 5

Pseudocódigo Norma 4
Asignaciones y Mensajes

- Si se requiere que en pantalla salga un mensaje y a continuación el contenido de una variable, se coloca el mensaje entre comillas y, luego de haberlas cerrado, el nombre de la variable de interés.

- Por ejemplo:

N = 8
Escriba “El valor es” N

Generará en pantalla: *El valor es 8*

Pseudocódigo Norma 4
Asignaciones y Mensajes

- Si se quiere mostrar en pantalla el contenido de varias variables, entonces simplemente a continuación de la orden “Escriba”, y separadas por comas, se escriben los nombres de las variables cuyos contenidos se requiere visualizar.
- Por ejemplo:

  N = 8
  M = 4
  Escriba “Los valores son” N, M

Escribirá en pantalla: *Los valores son* 8 4

Pseudocódigo Norma 5
Lecturas Usuario

Pseudocódigo Norma 5
Lecturas Usuario

- Cuando se requiere leer un dato del usuario para que sea almacenado en una variable determinada, se utiliza la orden "Lea". Por ejemplo:

```
Lea N
```

- Que resume: *Lea* un dato entero y guárdelo en la variable ***N*** que también es entera.

Pseudocódigo Norma 5
Lecturas Usuario

- Cuando se necesita leer más de un dato para ser almacenado en diferentes variables se utiliza la orden “Lea” y escribir las variables separadas por comas.

- Por ejemplo: **Lea** a,b

- Suponiendo que tanto ***a*** como ***b*** son variables de tipo entero, esta orden le indicará al computador que lea un dato entero y lo almacene en la variable ***a*** y luego que lea otro dato entero y lo almacene en la variable ***b***.

Pseudocódigo Norma 5
Lecturas Usuario

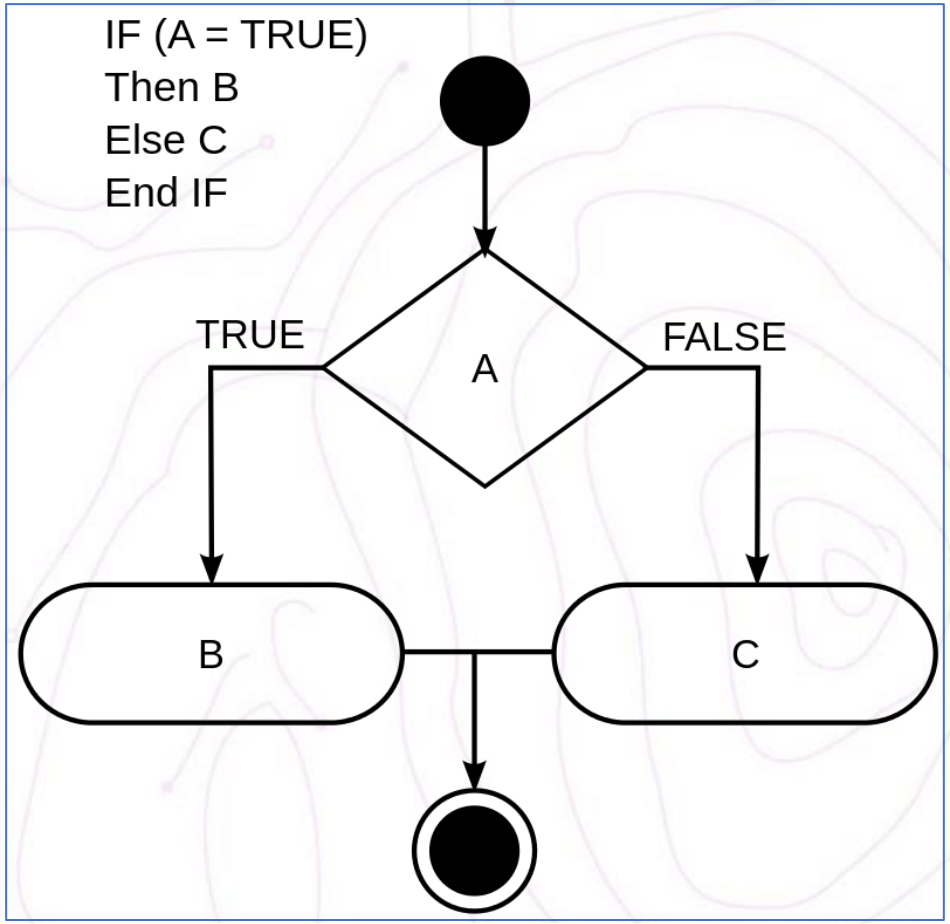
- No necesariamente cuando se lean dos o mas variables utilizando una sola orden “Lea”, estas deben ser del mismo tipo.

- Por ejemplo:

```
Lea var_e, var_r, var_c
```

- Asumiendo que ***var_e*** es una variable de tipo entero, ***var_r*** es una variable de tipo real y ***var_c*** es una variable de tipo carácter, esta orden le indicará al computador que lea valores de tipos diferentes y los almacene en las variables correspondientes.

Pseudocódigo Norma 6 Condicionales



Pseudocódigo Norma 6 Condicionales

- Cuando se necesita tomar una decisión, se utiliza la orden ***Si***, a continuación escribir la ***condición correspondiente*** y luego las instrucciones que se han de realizar en caso de que la condición sea ***Verdadera***.
- En caso de que la condición sea ***Falsa*** y tenga instrucciones a realizarse cuando así sea, entonces deberá existir una alternativa ***Si no***.
- Al finalizar toda la decisión deberá existir un indicador ***Fin_Si***.

Pseudocódigo Norma 6 Condicionales

```
Si  Condición
        .
        .
        Instrucciones a ejecutar si la condición es Verdadera
        .
        .
Sino
        .
        .
        Instrucciones a ejecutar si la condición es Falsa
        .
        .
Fin_Si
```

Condicionales – Operadores Relacionales

- Las condiciones pueden ser expresadas utilizando los siguientes ***Operadores Relacionales*** que son los símbolos que nos van a permitir obtener una respuesta ***Verdadera o Falsa***:

>	Mayor que
<	Menor que
>=	Mayor o igual a
<=	Menor o igual a

=	Igual a (también llamado igual de comparación)
< >	Diferente de

Condiciones – Proposiciones

- Una ***proposición*** es un ***enunciado declarativo*** que puede ser calificado sin ambigüedad como verdadero o falso.

- En este análisis no se tendrán en cuenta proposiciones que requieran una opinión individual y que por lo tanto, no pueden ser ***verdaderas o falsas***.

Condiciones – Proposiciones

1. Lógica y Algoritmos es una materia que hace parte del Programa de Ingeniería de Sistemas de la Universidad Libre Seccional Pereira.

2. El promedio a nivel nacional en el ECAES (Pruebas SaberPro) de Ingeniería de Sistemas fue de 110.3 puntos en el año 2007.

3. 5 es igual a 5.

4. *a* es mayor que *b*.

Condiciones – Proposiciones

Las siguientes ***NO*** son Proposiciones:

1. Atlético Nacional es el mejor equipo del fútbol colombiano.

2. Andrés Pastrana ha sido el mejor presidente de los colombianos.

3. Los próximos juegos deportivos nacionales serán ganados por el departamento del Risaralda.

Proposiciones: Primitivas y Compuestas

- Las proposiciones pueden considerarse como ***primitivas***, ya que en realidad no se pueden descomponer en partes más simples.

- Las proposiciones primitivas se utilizan con ***conectivos lógicos*** para formar proposiciones ***compuestas***.

- Los símbolos $p, q, r, s, \ldots$ se utilizan para denotar proposiciones, los cuales se llamarán variables proposicionales.

Conectivos Proposicionales

Los ***conectivos proposicionales*** son también conocidos con el nombre de ***conectivos lógicos***.

Los conectivos principales son:

- la **negación**, representada por el símbolo ~
- la **disyunción**, representada por el símbolo ∨
- la **conjunción**, representada por el símbolo ∧
- la **implicación**, representada por el símbolo →
- la **doble implicación**, representada por el símbolo ↔

Conectivos Proposicionales

Operador Lógico NOT o “Negación”:

- Este operador actúa sobre una sola expresión y lo que hace es que ***invierte*** el sentido de la Condición (que puede ser primitiva o compuesta).

- En otras palabras cuando el operador **NOT** va antes de una condición, entonces toda la expresión será ***Verdadera*** si deja de cumplirse la condición.

Tabla de Verdad - Negación

Dependiendo del lenguaje, la ***negación*** puede representarse con alguno de los siguientes símbolos:

- **not**
- **!**
- ¬
- ~

p	$\sim p$
V	F
F	V

Conectivos Proposicionales

Operador Lógico de Disyunción:

- Este operador lógico también se conoce como **OR**.

- Cuando el operador **OR** une dos condiciones, toda la expresión es ***Verdadera*** si, al menos, una de las dos es verdadera.

- Naturalmente, en el caso en que las dos condiciones (proposiciones) sean verdaderas, entonces todas la expresión será ***Verdadera***.

Tabla de Verdad - Disyunción

Dependiendo del lenguaje, la ***disyunción (o lógico)*** puede representarse con alguno de los siguientes símbolos:

- **or**
- ||

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

Conectivos Proposicionales

Operador Lógico de Conjunción:

- Este operador lógico también se conoce como **AND**, y solamente genera ***Verdadero*** si ambas condiciones se cumplen.

- No olvidar que se habla de ambas condiciones porque el operador **AND** puede “conectar” solamente dos condiciones y en cualquier otro caso genera ***Falso***.

Tabla de Verdad - Conjunción

Dependiendo del lenguaje, la ***conjunción (y lógico)*** puede representarse con alguno de los siguientes símbolos:

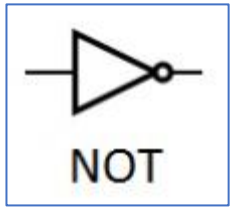
- **and**
- **&&**

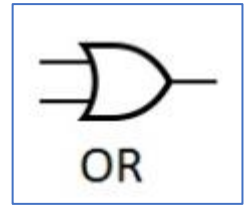
p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

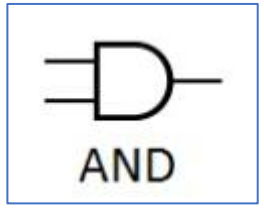
Conectivos Proposicionales

- Con estos ***tres operadores lógicos*** (también conocidos como operadores Booleanos) podemos construir una gran cantidad de decisiones y permitir que el computador las ejecute correctamente.

- Cabe anotar que evaluar una decisión y determinar si es ***Verdadera*** o ***Falsa*** es lo que más tiempo le toma a un computador (costo computacional), cuando dicha decisión está implementada en un Lenguaje de Programación.







Ejemplo 1

- Cuando éramos niños nos decían al mandarnos a la tienda:

"Tráigame una gaseosa y un pan de $1000"

Ejemplo 1

Como se trata de una ***proposición compuesta*** a partir de ***proposiciones primitivas*** relacionadas con un ***conectivo lógico***, se presentan varias opciones como se especificó en las tablas de verdad:

a) Si no traíamos ninguna de las dos cosas entonces no habíamos cumplido la orden.

b) Si no traíamos la gaseosa pero sí traíamos el pan de $100,oo tampoco habíamos cumplido la orden.

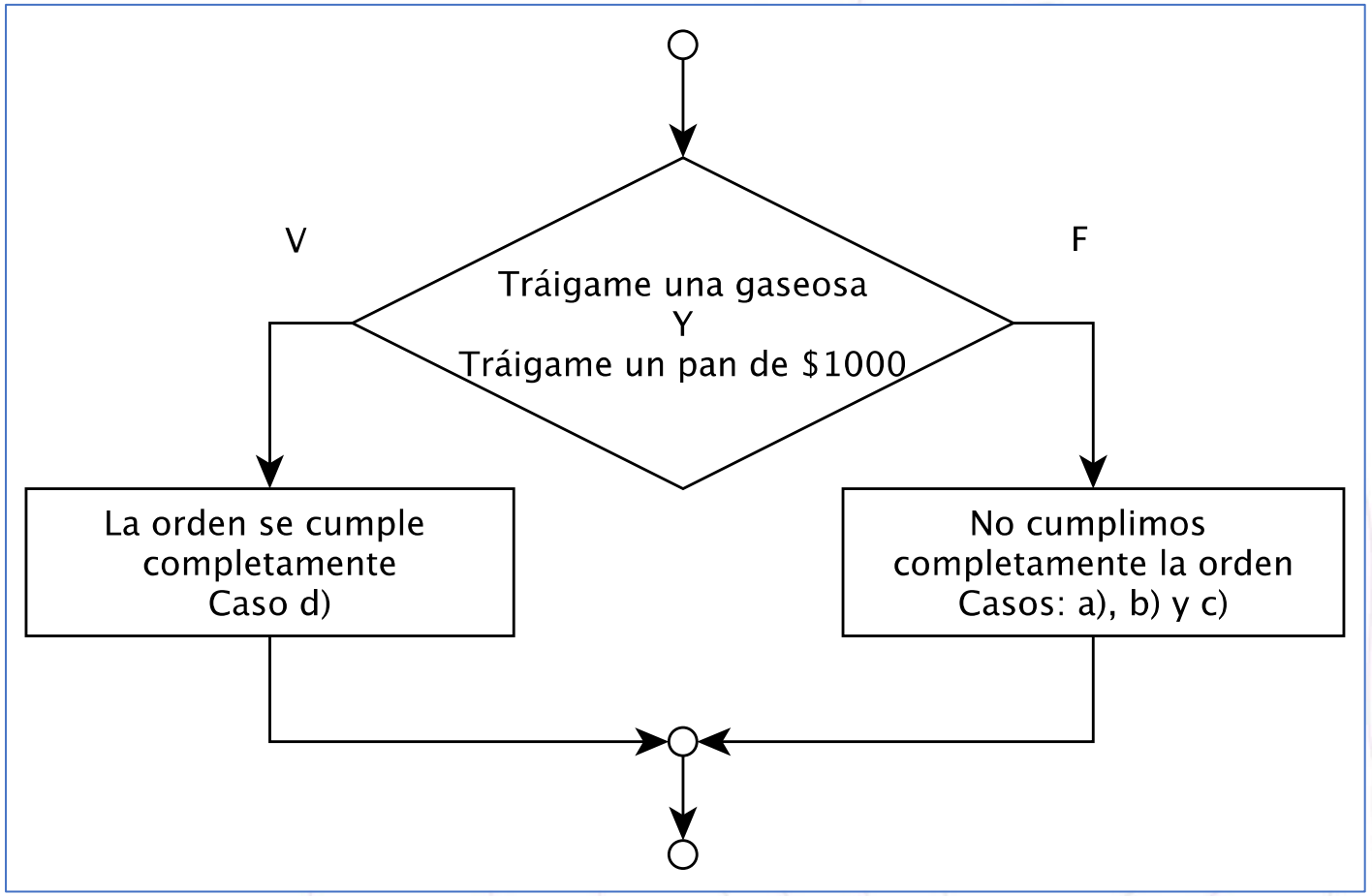
c) Si traíamos la gaseosa pero no traíamos el pan de $100,oo tampoco habíamos cumplido la orden.

d) Si traíamos la gaseosa y también traíamos el pan de $100,oo entonces allí sí habíamos cumplido la orden completamente.

Ejemplo 1

Tabla de Verdad Condicional Compuesto				
Caso	**Condición 1 Tráigame una Gaseosa**	**Condición 2 Tráigame un pan de $1000**	**Cond1 Y Cond2**	**Explicación Textual**
a)	F	F	F	No cumplimos la orden
b)	F	V	F	No cumplimos la orden
c)	V	F	F	No cumplimos la orden
d)	V	V	V	**Cumplimos la orden**

Ejemplo 1


V
F
Tráigame una gaseosa
Y
Tráigame un pan de $1000
La orden se cumple completamente
Caso d)
No cumplimos completamente la orden
Casos: a), b) y c)

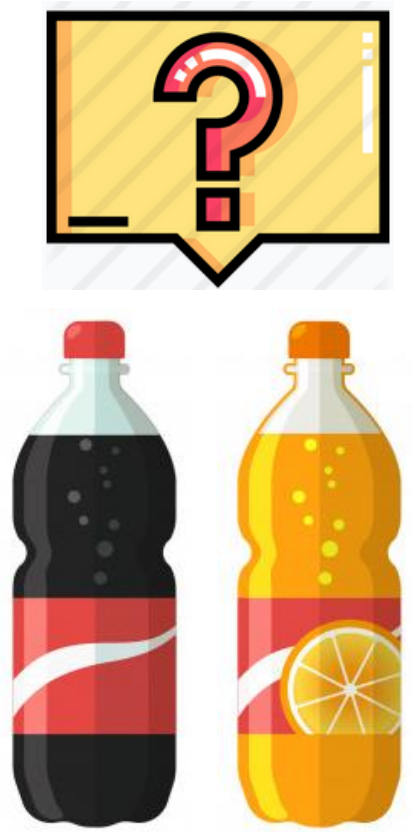
Ejemplo 1

```
1  Si Tráigame una gaseosa Y Tráigame un pan de $1000:
2      //Cuando la condición o proposición compuesta es Verdadera
3      La orden se cumple completamente// Caso d)
4  Sino
5      //Cuando la condición o proposición compuesta es Falsa
6      La orden no se cumple completamente// Casos a), b) y c)
7  Fin_Si
```

Ejemplo 2

- Cuando nos decían :

“Tráigame una Coca-Cola Litro o una Naranja Litro”



Ejemplo 2

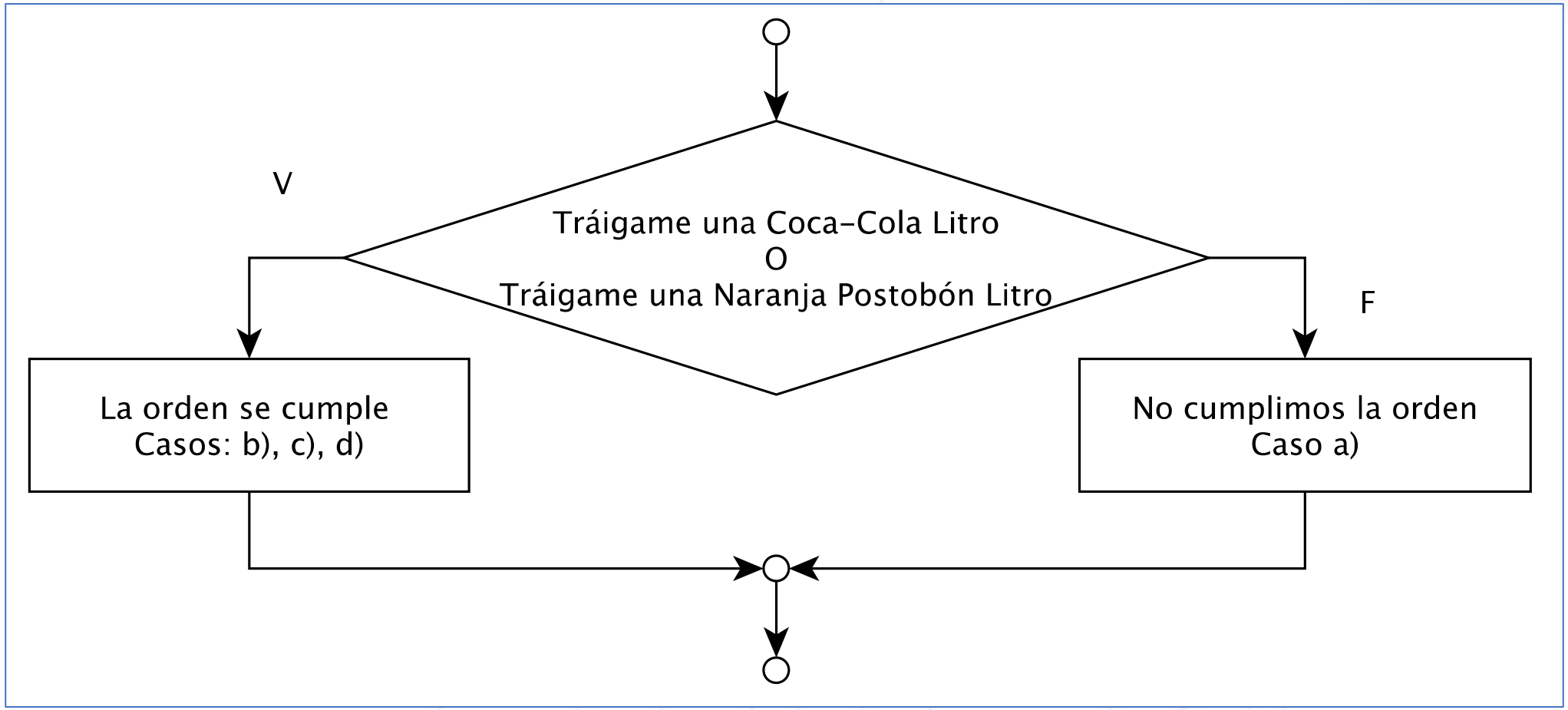
Se presentan varias opciones como se especificó en las tablas de verdad:

a) Si no traemos ninguna de las dos gaseosas entonces no hemos cumplido la orden
b) Si no traemos la Coca-Cola Litro y traemos la Naranja Litro entonces hemos cumplido la orden
c) Si traemos la Coca-Cola Litro y no traemos la Naranja Litro entonces hemos cumplido la orden
d) Si traemos ambas gaseosas hemos cumplido sobradamente la orden

Ejemplo 2

Tabla de Verdad Condicional Compuesto				
Caso	**Condición 1 Tráigame una Coca-Cola Litro**	**Condición 2 Tráigame un Naranja Postobón Litro**	**Cond1 O Cond2**	**Explicación Textual**
a)	F	F	F	No cumplimos la orden
b)	F	V	V	**Cumplimos la orden**
c)	V	F	V	**Cumplimos la orden**
d)	V	V	V	**La orden se cumple en ambos casos**

Ejemplo 2


V
Tráigame una Coca-Cola Litro
O
Tráigame una Naranja Postobón Litro
F
La orden se cumple
Casos: b), c), d)
No cumplimos la orden
Caso a)

Ejemplo 2

```
1  Si Tráigame una Coca-Cola Litro O Tráigame una Naranja Postobón Litro:
2      //Cuando la condición o proposición compuesta es Verdadera
3      La orden se cumple // Casos b), c) y d)
4  Sino
5      //Cuando la condición o proposición compuesta es Falsa
6      La orden no se cumple// Caso a)
7  Fin_Si
```

Ejemplo 3

```
A = 10
Si NOT( A  = 12 )
```

- En la primera línea estamos asignando el valor 10 a la variable A y en la segunda línea estamos preguntando que si A no es igual a 12, condición que es Verdadera debido a que la variable A es igual a 10.

Ejemplo 3

Si NOT (A > B) → Si (A < = B)

- ¿Cuándo A no es mayor que B?

  Cuando es Menor o Igual a B.

Pseudocódigo Norma 7
Ciclos

Pseudocódigo Norma 7
Ciclos

```
Mientras Condición Haga
        ..............
        ..................
        ..................
        Cuerpo del ciclo
        ..............
        ..................
Fin_Mientras
```

Pseudocódigo Norma 7
Ciclos

- En el Cuerpo del Ciclo se colocan las ordenes que se van a repetir (o iterar) mientras la condición sea ***Verdadera***.

- El **Fin_Mientras** le indicará posteriormente hasta dónde llega el bloque de instrucciones u órdenes y determinar a partir de dónde se devuelve el control del algoritmo para evaluar la condición.

Universidad Tecnológica de Pereira

Mision TIC 2022

El futuro digital es de todos

MinTIC

Algoritmos

Conceptualización, representación y codificación

Hechos QUE CONECTAN

Metodología, Técnica y **Tecnología** Para Solucionar un Problema

Solución de un Problema Mediante el Computador

a) Clarificación del Objetivo
b) Algoritmo
c) Prueba de Escritorio

1. Concepción del Problema

2. Técnicas de Representación

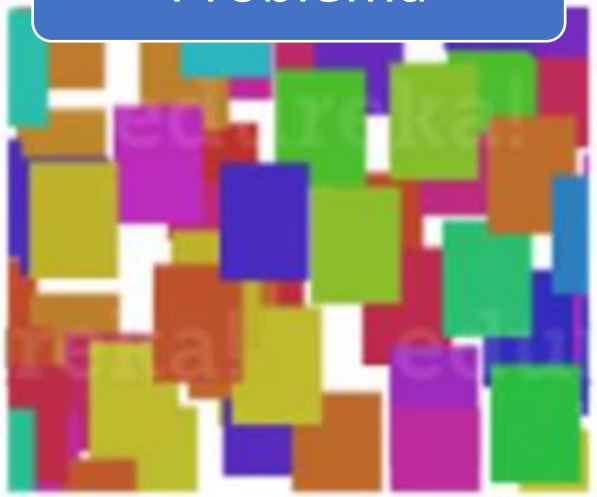
- Diagramas de Flujo
- Pseudocódigo

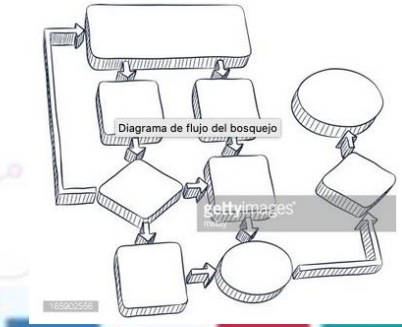
- Python
- Java
- Pascal
- C++
- Javascript

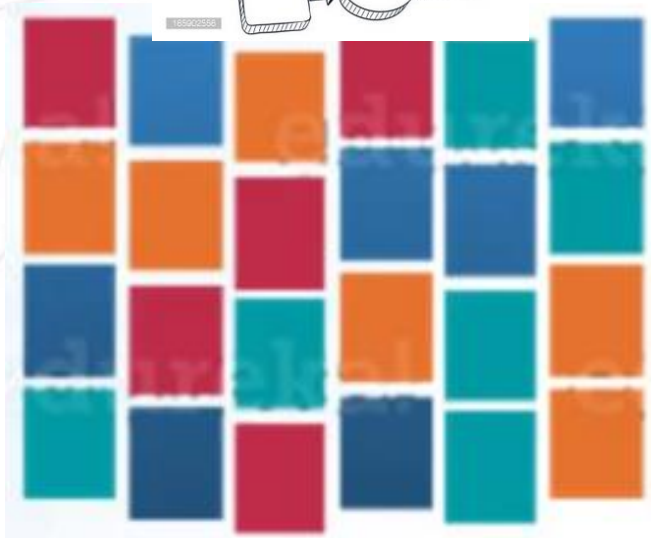
3. Transcripción o Codificación

Solución de un Problema Mediante el Computador

1. Concepción del Problema







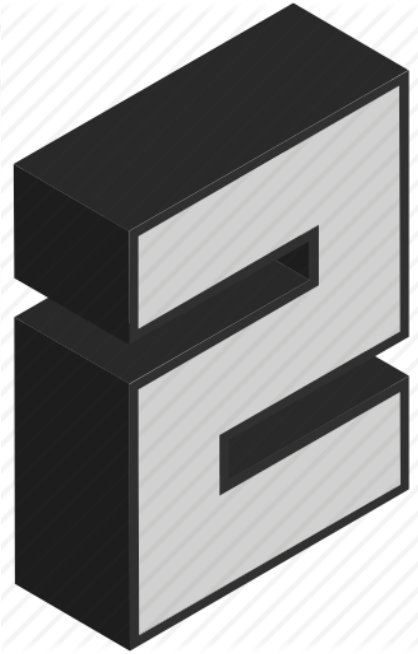
2. Técnicas de Representación

3. Transcripción o Codificación

Hechos QUE CONECTAN

Ejemplo 1
Concepción - Representación - Codificación

- Desarrollar un programa que permite leer un número entero positivo y determinar si es par.



Ejemplo 1
Concepción - Representación - Codificación

a) Clarificación del objetivo:

- Se trata de recibir un número entero (para lo cual utilizaremos una variable de tipo entero), verificar que es un número positivo y determinar si es un número par.

- Recordemos pues que son números pares aquellos que son divisibles exactamente entre dos, o sea, aquellos que al dividirlos entre 2 su residuo es cero.

Ejemplo 1
Concepción - Representación - Codificación

a) Clarificación del objetivo:

- En caso de que el número leído sea para avisaremos a través de un título que el número sí es par y en caso de que no sea así entonces haremos lo mismo avisando que el número no es par.

- Apenas hayamos avisado a través de un título que el número es par o que no lo es, entonces allí deberá terminar nuestro algoritmo.

Ejemplo 1

Concepción - Representación - Codificación

b) Algoritmo Informal:

```
1  //Algoritmo para determinar si un número es par
2  Inicio
3      Leer un número y guardarlo en una variable entera
4      Si ese número es negativo
5          Escribir que ese número no sirve para nuestro propósito
6      Sino
7          Preguntar si el número es par
8          Si lo es entonces escribir que el número leído es par
9          Si no lo es escribir que el número leído no es par
10 Fin
```

Ejemplo 1

Concepción - Representación - Codificación

b) Algoritmo Informal:

```
1  //Algoritmo para determinar si un número es par
2  Inicio
3      Leer un número y guardarlo en una variable entera
4      Si ese número es negativo
5          Escribir que ese número no sirve para nuestro propósito
6      Sino
7          Preguntar si el número es par
8          Si lo es entonces escribir que el número leído es par
9          Si no lo es escribir que el número leído no es par
10 Fin
```

Ejemplo 1

Concepción - Representación - Codificación

b) Algoritmo (Estandarizado):

```
1  //Algoritmo Número_Par
2  Variables
3      Entero: num
4  Inicio
5      Lea num
6      Si num < 0
7          Escriba “El número debe ser positivo”
8      Sino
9          Si num / 2 * 2 = num
10             Escriba “El número leído es par”
11         Sino
12             Escriba “El número leído no es par”
13 Fin
```

Ejemplo 1

Concepción - Representación - Codificación

c) Prueba de Escritorio:

PANTALLA

MEMORIA

num

Ejemplo 1
Concepción - Representación - Codificación

c) Prueba de Escritorio:

```
num / 2 * 2  =  num
18  / 2 * 2  =  18
    9   * 2  =  18
        18   =  18
```

(División entera)

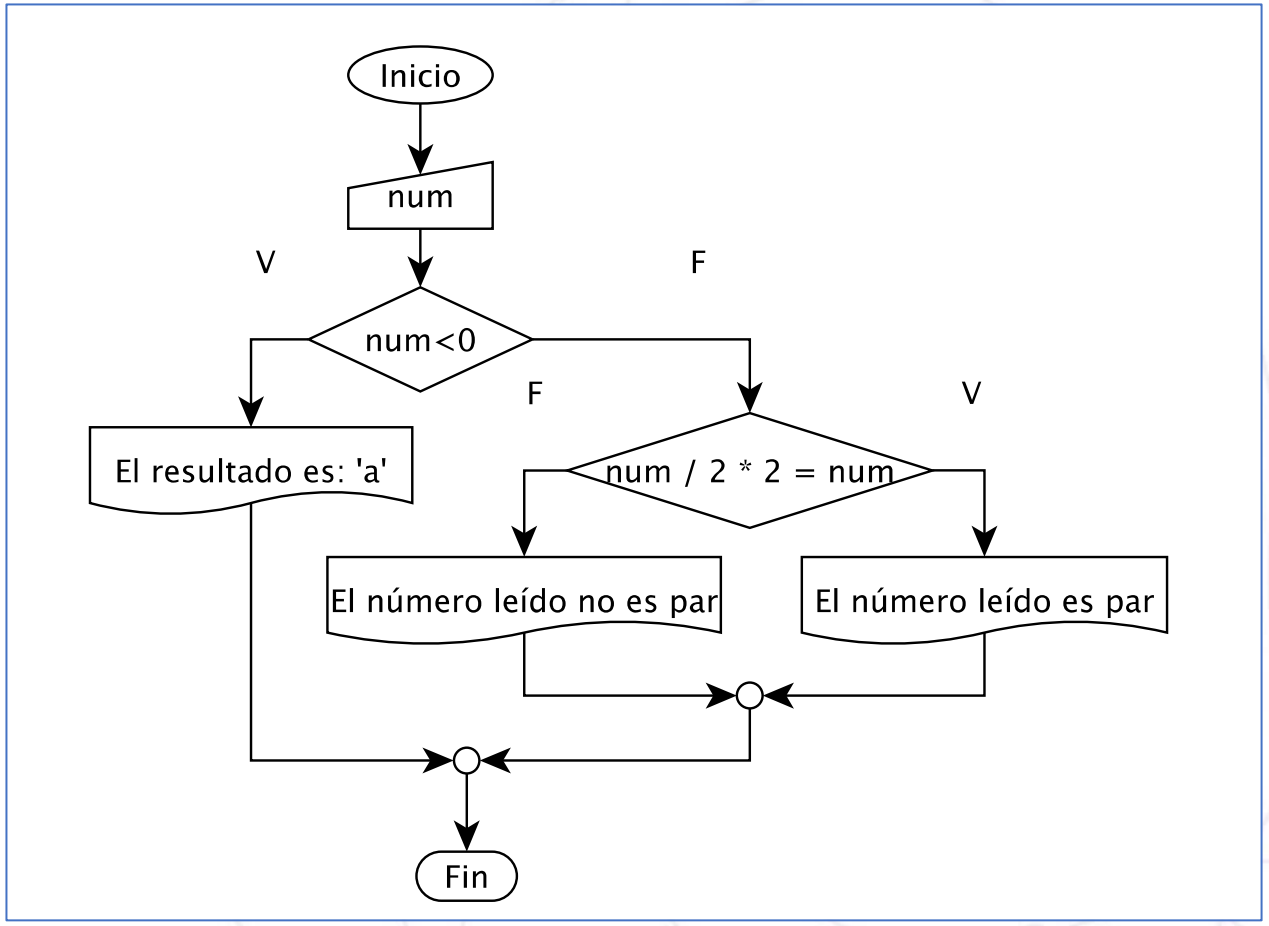
Ejemplo 1

Concepción - Representación - Codificación

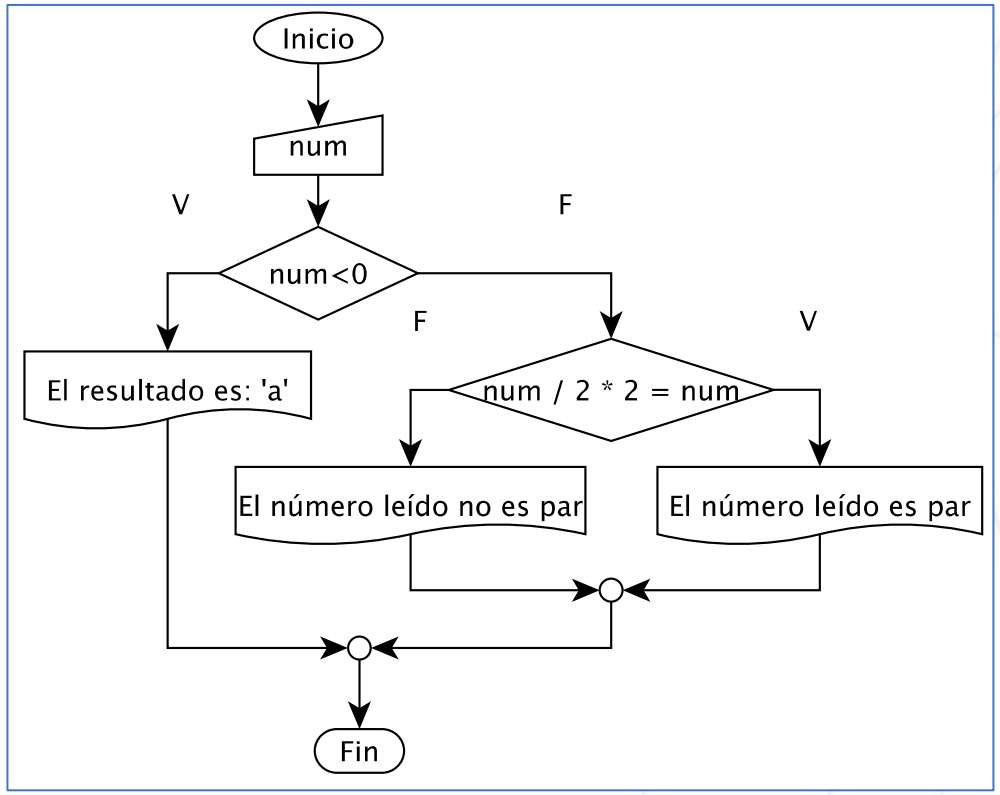
c) Prueba de Escritorio:

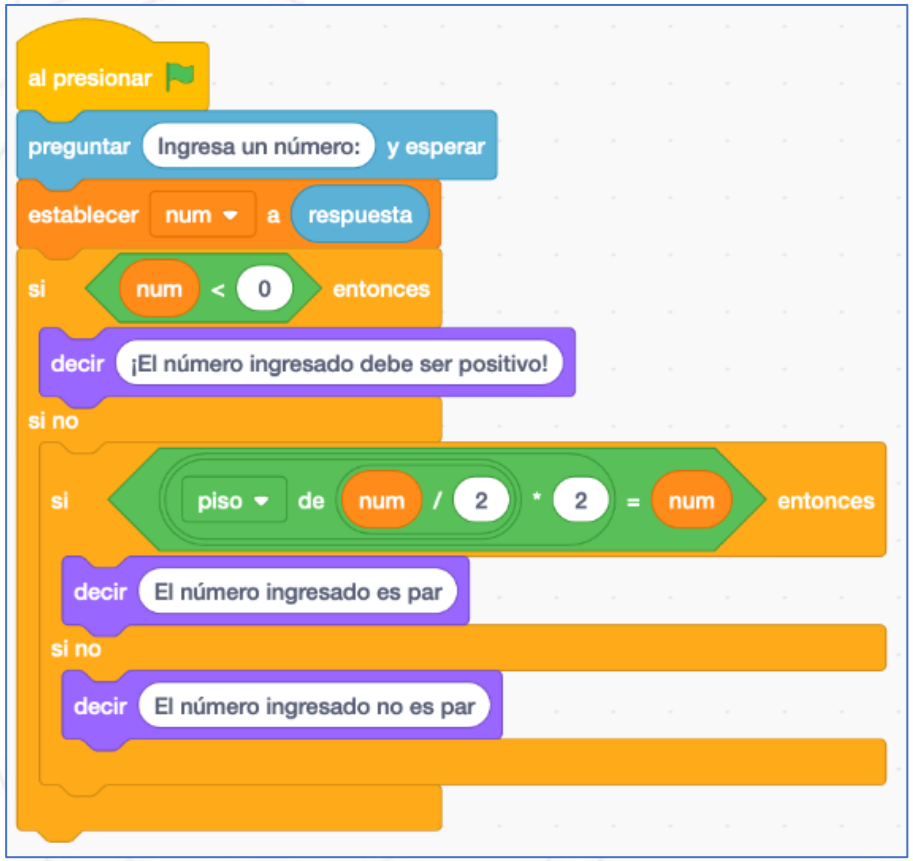
PANTALLA	MEMORIA
18 El número leído es par	num 18

Ejemplo 1
Concepción - Representación - Codificación



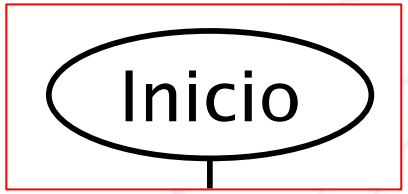
Ejemplo 1
Concepción - Representación - Codificación





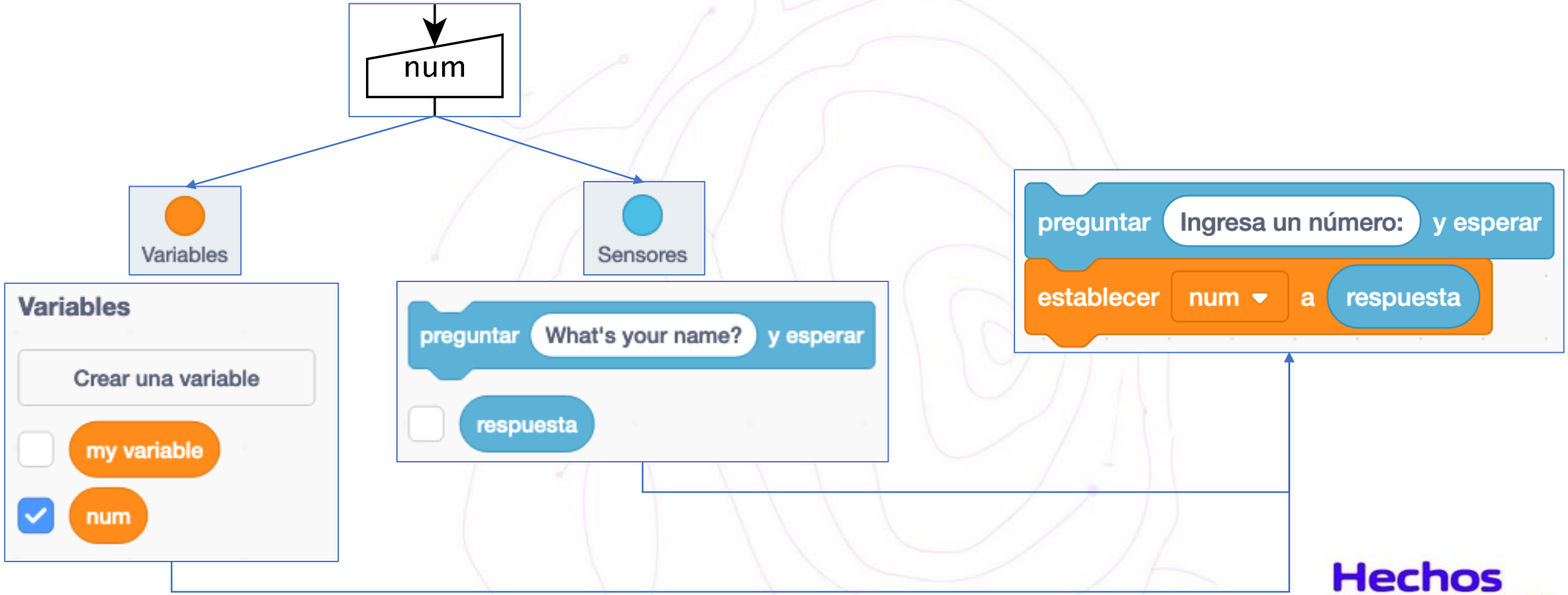
Enlace Implementación: https://scratch.mit.edu/projects/518638680

Ejemplo 1
Concepción - Representación - Codificación



Enlace Implementación: https://scratch.mit.edu/projects/518628680

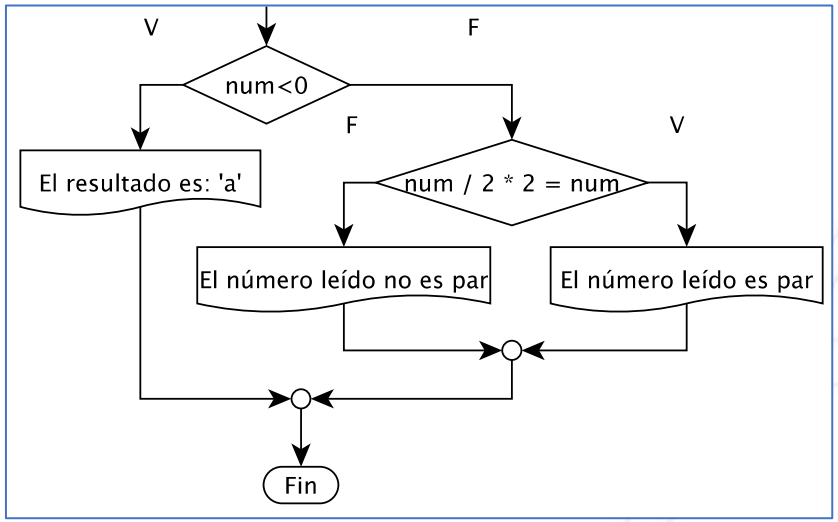
Ejemplo 1
Concepción - Representación - Codificación

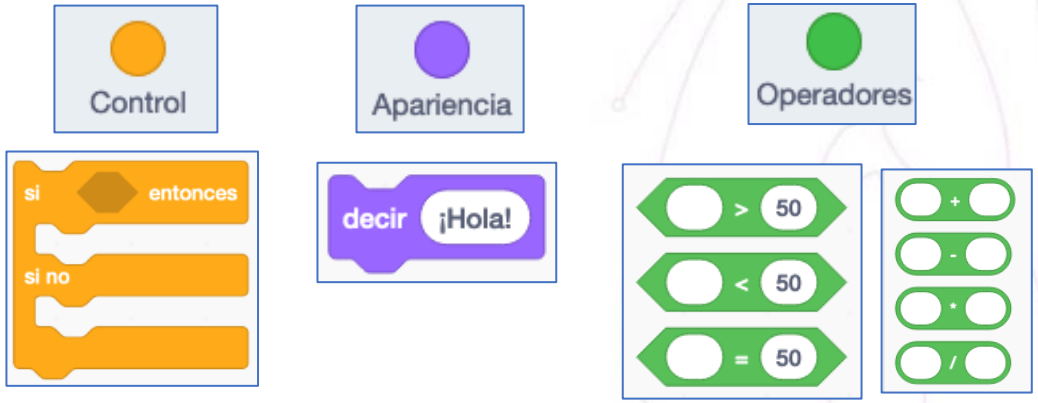


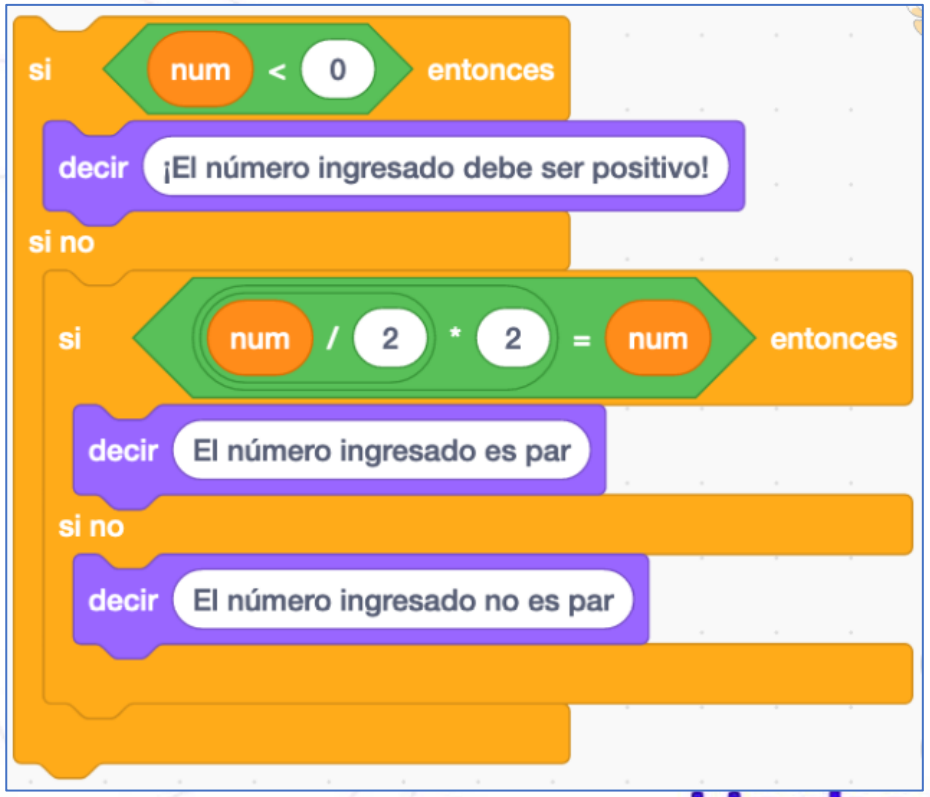
Enlace Implementación: https://scratch.mit.edu/projects/518638680

Ejemplo 1

Concepción - Representación - Codificación







Enlace Implementación: https://scratch.mit.edu/projects/518638680

Ejemplo 1
Concepción - Representación - Codificación

```
//Algoritmo Número_Par
Variables
    Entero: num
Inicio
    Lea num
    Si num < 0
        Escriba “El número debe ser positivo”
    Sino
        Si num / 2 * 2 = num
            Escriba “El número leído es par”
        Sino
            Escriba “El número leído no es par”
Fin
```

Microsoft Visual Basic

main.vb

```
Module VBModule
    Sub Main()
        input num
        if num < 0 then
            print “El número debe ser positivo”
        else
            if int(num/2*2) = num then
                print “El numero leído es par”
            else
                print “El número leído no es par”
    End Sub
End Module
```

Ejemplo 1
Concepción - Representación - Codificación

```
1  //Algoritmo Número_Par
2  Variables
3      Entero: num
4  Inicio
5      Lea num
6      Si num < 0
7          Escriba “El número debe ser positivo”
8      Sino
9          Si num / 2 * 2 = num
10             Escriba “El número leído es par”
11         Sino
12             Escriba “El número leído no es par”
13 Fin
```

PASCAL

```
1  program numero_par;
2  var
3      num : integer;
4  begin
5      readln(num);
6      if (num < 0) then
7          writeln(‘ El número debe ser positivo ‘);
8      else
9          if (num/2*2 = num) then
10             writeln(‘ El número leído es par ‘);
11         else
12             writeln(‘ El número leído no es par ‘);
13 end
```

Ejemplo 1
Concepción - Representación - Codificación

```
1  //Algoritmo Número_Par
2  Variables
3      Entero: num
4  Inicio
5      Lea num
6      Si num < 0
7          Escriba “El número debe ser positivo”
8      Sino
9          Si num / 2 * 2 = num
10             Escriba “El número leído es par”
11         Sino
12             Escriba “El número leído no es par”
13 Fin
```

```
1  #include <iostream.h>
2
3  void main(){
4      int num;
5
6      cin >> num;
7      if ( num < 0)
8          cout << “El número debe ser positivo “;
9      else
10         if (num/2*2 == num)
11             cout << “El número leído es par”;
12         else
13             cout << “El número leído no es par”;
14 }
```

Ejemplo 1
Concepción - Representación - Codificación

```
1  //Algoritmo Número_Par
2  Variables
3      Entero: num
4  Inicio
5      Lea num
6      Si num < 0
7          Escriba “El número debe ser positivo”
8      Sino
9          Si num / 2 * 2 = num
10             Escriba “El número leído es par”
11         Sino
12             Escriba “El número leído no es par”
13 Fin
```

COBOL

```
1  IDENTIFICATION DIVISION.
2  PROGRAM_ID. NUMERO_PAR.
3  ENVIRONMENT DIVISION.
4  CONFIGURATION SECTION.
5   SOURCE-COMPUTER. CLON.
6  OBJECT-COMPUTER. CLON.
7
8   DATA DIVISION.
9  WORKING-STORAGE
10 SECTION. 01 NUM PIC 99.
11
12 PROCEDURE DIVISION.
13  INICIO.
14      ACCEPT NUM LINE 10 POSITION 10 NO BEEP
15      IF NUM IS LESS THAN 0 THEN
16          DISPLAY “ EL NÚMERO DEBE SER POSITIVO” LINE 12 COL 10
17      ELSE
18          IF NUM / 2 * 2 IS EQUAL TO NUM THEN
19              DISPLAY “EL NÚMERO LEÍDO ES PAR“ LINE 12 COL 10
20          ELSE
21              DISPLAY “EL NÚMERO LEÍDO NO ES PAR“ LINE 12 COL 10
22      STOP RUN.
```

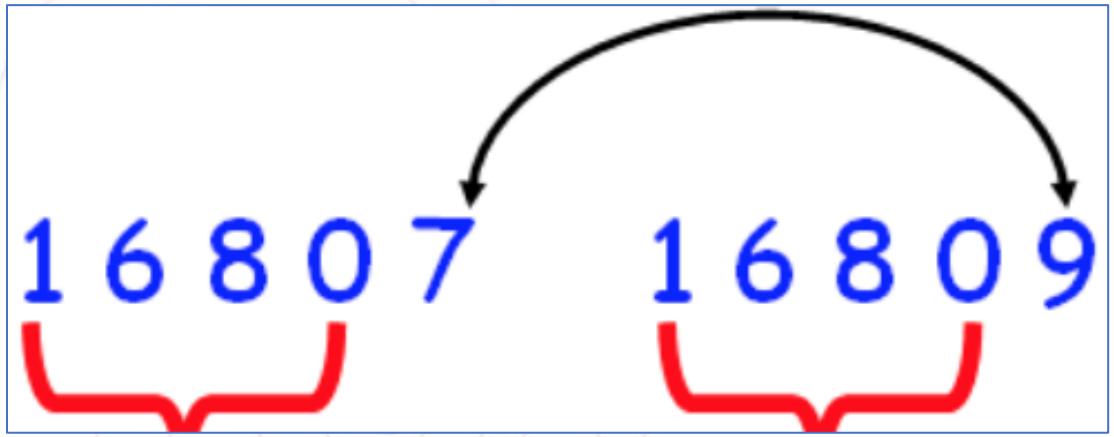
Ejemplo 1
Concepción - Representación - Codificación

```
1  //Algoritmo Número_Par
2  Variables
3      Entero: num
4  Inicio
5      Lea num
6      Si num < 0
7          Escriba “El número debe ser positivo”
8      Sino
9          Si num / 2 * 2 = num
10             Escriba “El número leído es par”
11         Sino
12             Escriba “El número leído no es par”
13 Fin
```

```
1  num = int(input())
2  if num < 0:
3      print("El número debe ser positivo")
4  else:
5      if int(num/2)*2 == num:
6          print("El número es par")
7      else:
8          print("El número no es par")
9
```

Ejemplo 2
Concepción - Representación - Codificación

- Leer dos números enteros positivos y determinar si el último dígito de un número es igual al último dígito del otro.



Ejemplo 2

Concepción - Representación - Codificación

```
Algoritmo para comparar el último dígito
Inicio
        Leer un número entero y guardarlo en una variable entera
        Leer otro número entero y guardarlo en otra variable entera

        Guardar en una variable el último dígito del primer número leído
        Guardar en otra variable el último dígito del último dígito leído


        Comparar el contenido de estas dos últimas variables
        Si son iguales
                Escribir que los dos últimos dígitos son iguales
        Si no son iguales
                Escribir que los dos últimos dígitos no son iguales
Fin
```

Ejemplo 2

Concepción - Representación - Codificación

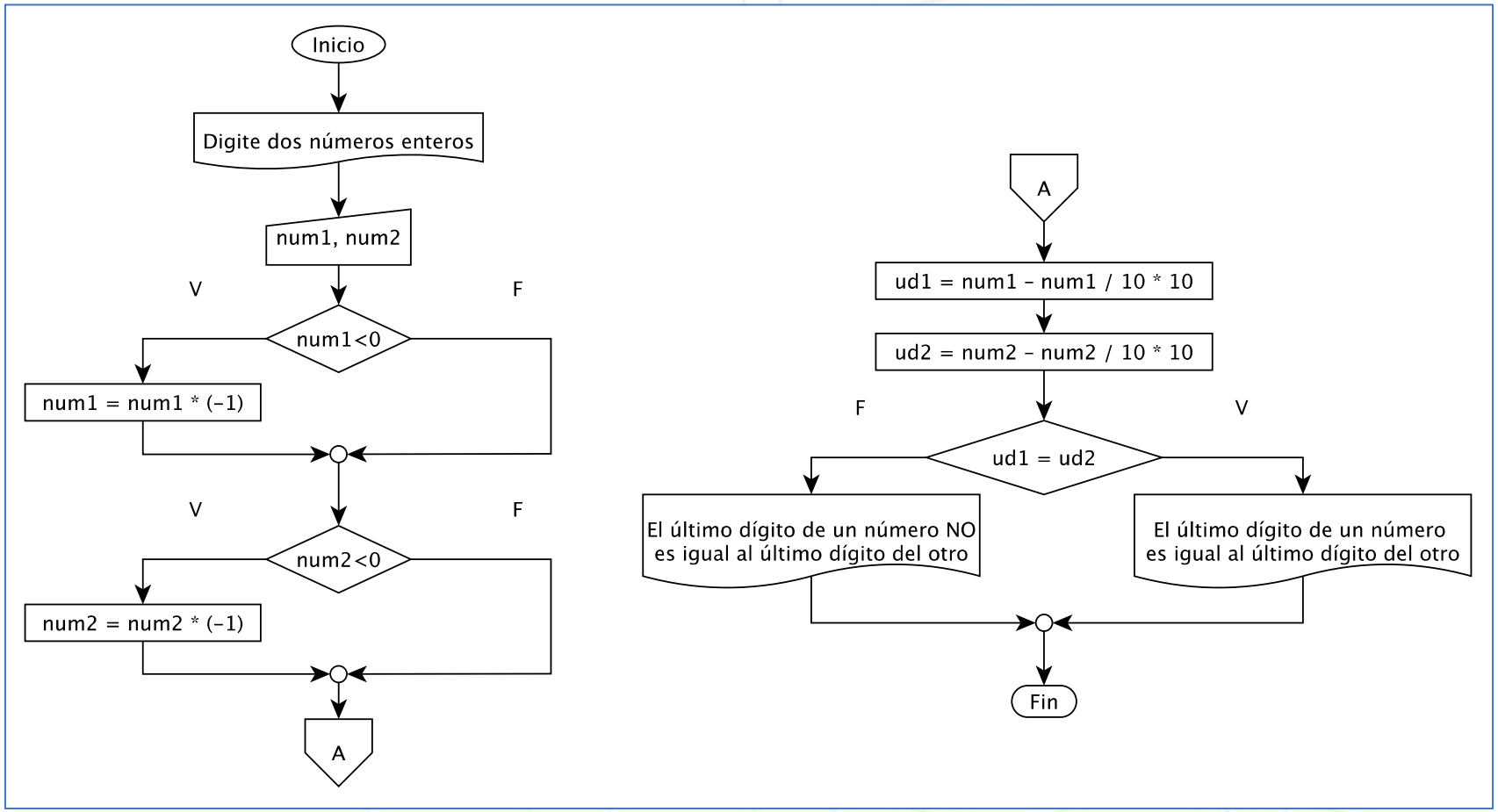
$$
\begin{aligned}
ud &= num - num / 10 * 10 \\
ud &= 854 - 854 / 10 * 10 \\
ud &= 854 - 85 * 10 \\
ud &= 854 - 850 \\
ud &= 4
\end{aligned}
$$

Ejemplo 2
Concepción - Representación - Codificación

```
Algoritmo Compara_Ult_digs
Variables
        Entero :  num1, num2, ud1, ud2
Inicio
        Escriba "Digite dos números enteros "
        Lea num1, num2

        Si num1 < 0
                num1 = num1 * (-1)
        Si num2 < 0
                num2 = num2 * (-1)

        ud1 = num1 – num1 / 10 * 10
        ud2 = num2 – num2 / 10 * 10

        Si ud1 = ud2
                Escriba "El último dígito de un número es igual al último dígito del otro"
        Sino
                Escriba "El último dígito de un número no es igual al último dígito del otro"

Fin
```

Ejemplo 2

Concepción - Representación - Codificación



Ejemplo 2
Concepción - Representación - Codificación

Enlace Implementación:
https://scratch.mit.edu/projects/518653042

Bibliografía

- [1] Trejos, O. (2000). La esencia de la lógica de programación. *Pereira, Colombia: Centro Editorial Universidad de Caldas*.
- [2] Hehner, E. C. (1984). *The logic of programming*. Prentice/Hall International.
- [3] Gabbrielli, M., & Martini, S. (2010). Programming languages: principles and paradigms. Springer Science & Business Media.
- [4] Marji, M. (2014). Learn to program with Scratch: A visual introduction to programming with games, art, science, and math. No Starch Press.
- [5] Fabrizio Romano. (2018). *Learn Python Programming : The No-nonsense, Beginner's Guide to Programming, Data Science, and Web Development with Python 3.7, 2nd Edition: Vol. 2nd ed*. Packt Publishing.