1. ALGORITMOS

El sistema para describir (“escribir”) un algoritmo consiste en realizar una descripción paso a paso con un lenguaje natural del citado algoritmo. Recordemos que un algoritmo es un método o conjunto de reglas para solucionar un problema. En cálculos elementales estas reglas tienen las siguientes propiedades:

- Deben ir seguidas de alguna secuencia definida de pasos hasta que se obtenga un resultado coherente,
- Sólo puede ejecutarse una operación a la vez.
- Todo algoritmo tiene la siguiente forma:

- **ENTRADA:** son todas las condiciones necesarias para cumplir el objetivo
- **PROCESO:** El paso a paso del algoritmo de forma secuencial
- **SALIDA:** El resultado del algoritmo

Para diseñar un algoritmo se debe de tener en cuenta:

- El problema
- Objetivo.
- La sintaxis del Algoritmo

EL PROBLEMA: El computador es una máquina que por sí sola no puede hacer nada, necesita ser programada, es decir, introducirle instrucciones u ordenes que le digan lo que tiene que hacer. Un programa es la solución a un problema inicial, así que todo comienza allí: en el **PROBLEMA**. El proceso de programación es el siguiente: Dado un determinado problema el programador debe idear una solución y expresarla usando un algoritmo (aquí es donde entra a jugar); luego de esto, debe codificarlo en un determinado lenguaje de programación y por último ejecutar el programa en el computador el cual refleja una solución al problema inicial. Esto es a grandes rasgos lo que hace el **programador de computadores**.

OBJETIVO: aquel que nos permite saber qué es lo que se quiere alcanzar, por ejemplo, Es evidente que no podemos avanzar hacia la casa de un amigo nuestro que no sabemos en donde vive porque las posibilidades de que lleguemos son casi nulas.

SINTAXIS DE UN ALGORITMO

Inicio
    Procedimiento
Fin

Todo algoritmo debe de estar indentado "significa mover un bloque de texto hacia la derecha insertando espacios o tabuladores, para así separarlo del margen izquierdo y mejor distinguirlo del texto adyacente; en el ámbito de la imprenta, este concepto siempre se ha denominado sangrado o sangría." (Wikipedia,2019)

1.1 Algoritmos informales

Como se mencionó en la terminología esta clase de algoritmos no pueden ser ejecutados por los computadores y por lo tanto son preferiblemente realizables por el ser humano.

Ejemplo:

1. Realizar un algoritmo que me permita empacar un regalo.

Objetivo: Empacar un regalo
Algoritmo

Inicio

1. Se coloca el regalo encima del papel regalo
2. Se realiza el pliegue ubicado adelante
3. Se realiza el pliegue ubicado atrás
4. Se unen los dos pliegues
5. Se pegan los dos pliegues con la cinta
6. Se realiza un triángulo en el extremo derecho
7. Se pega con la cinta
8. Se realiza un triángulo en el extremo izquierdo
9. Se pega con la cinta
10. Se pega el moño con la cinta
11. Se marca la tarjeta
12. Se coloca la tarjeta con cinta al lado del moño.

Fin

consideremos el algoritmo que responde a la pregunta: ¿Qué hacer para ver la película de Harry Potter? La respuesta es muy sencilla y puede ser descrita en forma de algoritmo general de modo similar a:

1. ir al cine
2. comprar una entrada (billete o ticket)
3. ver la película
4. regresar a casa

El algoritmo consta de cuatro acciones básicas, cada una de las cuales debe ser ejecutada antes de realizar la siguiente. En términos de computadora, cada acción se codificará en una o varias sentencias que ejecutan una tarea particular. El algoritmo descrito es muy sencillo; sin embargo, el algoritmo general se descompondrá en pasos más simples en un procedimiento denominado **refinamiento sucesivo**, ya que cada acción puede descomponerse a su vez en otras acciones

simples. Así, por ejemplo, un primer refinamiento del algoritmo ir al cine se puede describir de la forma siguiente:

---

1. Ir al Cine

---

1. Inicio
2. ver la cartelera de cines en el periódico
3. **si** no proyectan "Harry Potter" **entonces**
   3.1. decidir otra actividad
   3.2. bifurcar al paso 7
   si_no
   3.3. ir al cine
   fin_si

---

2. Comprar la entrada

---

4. **si** hay cola entonces
   4.1. ponerse en ella
   4.2 **mientras** haya personas delante **hacer**
   4.2.1 avanzar en la cola
   Fin_mientras
   fin_si
5. **si** hay localidades **entonces**
   5.1. comprar una entrada
   5.2. pasar a la sala
   5.3. localizar la(s) butaca(s)
   5.4. **mientras** proyectan la película **hacer**

---

3. Ver la película

   5.4.1. ver la película
   fin_mientras
   5.5. abandonar el cine
   si_no
   5.6. refunfuñar
   fin_si

---

4. Regresar a casa

---

6. volver a casa

---

7. fin

En el algoritmo anterior existen diferentes aspectos a considerar. En primer lugar, ciertas palabras reservadas se han escrito deliberadamente en negrita (**mientras, si_no;** etc.). Estas palabras describen las estructuras de control fundamentales y procesos de toma de decisión en el algoritmo. Éstas incluyen los conceptos importantes de selección (expresadas por **si-entonces-si_no, if-then-else**) y de repetición (expresadas con **mientras-hacer o a veces repetir-hasta e iterar-fin_iterar, en**

inglés, while-do y repeat-until) que se encuentran en casi todos los algoritmos, especialmente en los de proceso de datos. La capacidad de decisión permite seleccionar alternativas de acciones a seguir o bien la repetición una y otra vez de operaciones básicas. (JOYANES, 2008)

Como se observa en el algoritmo anterior, este se encuentra indentado (sangrado o justificación)

1.2 REPRESENTACIÓN GRÁFICA DE LOS ALGORITMOS

Para representar un algoritmo se debe utilizar algún método que permita independizar dicho algoritmo del lenguaje de programación elegido. Ello permitirá que un algoritmo pueda ser codificado indistintamente en cualquier lenguaje. Para conseguir este objetivo se precisa que el algoritmo sea representado gráfica o numéricamente, de modo que las sucesivas acciones no dependan de la sintaxis de ningún lenguaje de programación, sino que la descripción pueda servir fácilmente para su transformación en un programa, es decir, su codificación. Los métodos usuales para representar un algoritmo son:

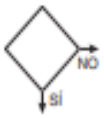
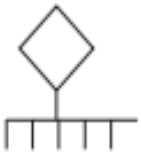
1. Diagrama de flujo
2. Lenguaje de especificación de algoritmos: pseudocódigo

1.2.1 Diagramas de flujo

Un diagrama de flujo (flowchart) es una de las técnicas de representación de algoritmos más antigua y a la vez más utilizada, aunque su empleo ha disminuido considerablemente, sobro todo, desde la aparición de lenguajes de programación estructurados. Un diagrama de flujo es un diagrama que utiliza los símbolos (cajas) estándar que tiene los pasos de algoritmo escritos en esas cajas unidas por flechas, denominadas líneas de flujo, que indican la secuencia en que se debe ejecutar. Los símbolos estándar normalizados por ANSI (abreviatura de American National Standars Institute) son muy variados. Sin embargo, los símbolos más utilizados representan:

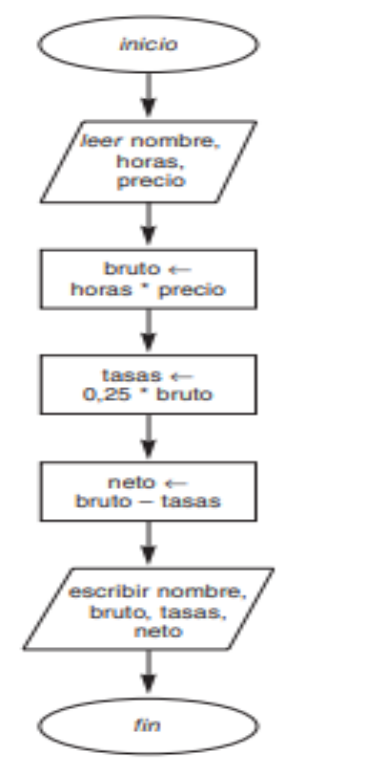
- proceso
- decisión
- Conectores
- fin
- entrada/salida
- dirección del flujo

Simbología

Símbolos principales	Función
	Terminal (representa el comienzo, "inicio", y el final, "fin" de un programa. Puede representar también una parada o interrupción programada que sea necesario realizar en un programa.
	Entrada/Salida (cualquier tipo de introducción de datos en la memoria desde los periféricos, "entrada", o registro de la información procesada en un periférico, "salida".
	Proceso (cualquier tipo de operación que pueda originar cambio de valor, formato o posición de la información almacenada en memoria, operaciones aritméticas, de transferencia, etc.).
	Decisión (indica operaciones lógicas o de comparación entre datos —normalmente dos— y en función del resultado de la misma determina cuál de los distintos caminos alternativos del programa se debe seguir; normalmente tiene dos salidas —respuestas SÍ o NO— pero puede tener tres o más, según los casos).
	Decisión múltiple (en función del resultado de la comparación se seguirá uno de los diferentes caminos de acuerdo con dicho resultado).
	Conector (sirve para enlazar dos partes cualesquiera de un ordinograma a través de un conector en la salida y otro conector en la entrada. Se refiere a la conexión en la misma página del diagrama.
	Indicador de dirección o línea de flujo (indica el sentido de ejecución de las operaciones).
	Línea conectora (sirve de unión entre dos símbolos).
	Conector (conexión entre dos puntos del organigrama situado en páginas diferentes).
	Llamada a subrutina o a un proceso predeterminado (una subrutina es un módulo independientemente del programa principal, que recibe una entrada procedente de dicho programa, realiza una tarea determinada y regresa, al terminar, al programa principal).
	Pantalla (se utiliza en ocasiones en lugar del símbolo de E/S).
	Impresora (se utiliza en ocasiones en lugar del símbolo de E/S).
	Teclado (se utiliza en ocasiones en lugar del símbolo de E/S).
	Comentarios (se utiliza para añadir comentarios clasificadores a otros símbolos del diagrama de flujo. Se pueden dibujar a cualquier lado del símbolo).

Ejemplo:

Problema: Calcular el salario bruto y el salario neto de un trabajador “por horas” conociendo el nombre, número de horas trabajadas, impuestos a pagar y salario neto



1.2.2 Seudocódigo

El pseudocódigo es un lenguaje de especificación (descripción) de algoritmos. El uso de tal lenguaje hace el paso de codificación final (esto es, la traducción a un lenguaje de programación) relativamente fácil. Se considera un primer borrador, dado que el pseudocódigo tiene que traducirse posteriormente a un lenguaje de programación. El pseudocódigo no puede ser ejecutado por una computadora. La ventaja del pseudocódigo es qué en su uso, en la planificación de un programa, el programador se puede concentrar en la lógica y en las estructuras de control y no preocuparse de las reglas de un lenguaje específico. Es también fácil modificar el pseudocódigo si se descubren errores o anomalías en la lógica del programa, mientras que en muchas ocasiones suele ser difícil el cambio en la lógica, una vez que está codificado en un lenguaje de programación. Otra ventaja del pseudocódigo es que puede ser traducido fácilmente a lenguajes como Java, JavaScript, Python, C char#, Php, etc. La escritura de pseudocódigo exige normalmente la indentación (sangría en el margen izquierdo) de diferentes líneas. Una representación en pseudocódigo —en inglés— de un problema de cálculo del salario neto de un trabajador es la siguiente:

```
Inicio
 //cálculo de impuesto y salarios
    Leer nombre, horas, precio
    salario ← horas * precio
    tasas ← 0,25 * salario
    salario_neto ← salario – tasas
    Escribir nombre, salario, tasas, salario
fin
```

El algoritmo comienza con la palabra inicio y finaliza con la palabra fin. Entre estas palabras, sólo se escribe una instrucción o acción por línea. La línea precedida por // se denomina comentario. Es una información al lector del programa y no realiza ninguna instrucción ejecutable, sólo tiene efecto de documentación interna del programa. Algunos autores suelen utilizar corchetes o llaves.

1.3 Algoritmos Computacionales

Se consideran como tales todos aquellos algoritmos que deben ser preferiblemente implementados en un computador para aprovechar su velocidad de procesamiento. Un ejemplo de estos puede ser el algoritmo que genere los primeros 100 números primos, recordando que un número primo es aquel que solo puede ser dividido exactamente entre la unidad y entre sí mismo, que, si bien podrían ser calculados utilizando un papel y un lápiz, la utilización de un computador en unión con el algoritmo adecuado nos va a dar un resultado mucho más rápido y absolutamente confiable (de hecho, depende de que el algoritmo igualmente sea muy confiable).

En el desarrollo de los algoritmos computacionales, los cuales nos van a ocupar en lo sucesivo, la metodología para llegar a la solución final que permita lograr un objetivo (igualmente computacional) continúa con los siguientes pasos:

- **Digitación:** Es el proceso a través del cual le escribimos al computador el programa que hemos acabado de escribir en papel. Para ello nos valemos de un programa llamado Editor de texto que nos permite escribir un texto y grabarlo. Visto neutralmente, un programa no es más que un texto escrito bajo la óptica de algunas reglas preestablecidas por los creadores de un Lenguaje de Programación.

- **Transcripción:** Este es el proceso a través del cual "convertimos" un algoritmo, escrito en términos muy coloquiales e informales, en un listado de instrucciones entendibles a un computador y que se ajustan a las reglas sintácticas de determinado lenguaje de programación. Podríamos decir que es la "traducción" de un algoritmo con la "ortografía" de un Lenguaje de Programación.

- **Prueba de escritorio**: ES el paso a paso de un algoritmo a través del papel para verificar que si cumple con el objetivo

- **Compilación:** 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. ¿Quién realiza realmente el proceso llamado compilación...? Pues lo realiza un programa llamado Compilador que es el encargado de evaluar dos tipos de errores:

    - **Errores de sintaxis:** Son aquellos errores representados en la omisión de alguna o algunas reglas sintácticas (hablando de un Lenguaje de Programación). Por ejemplo, es normal que algunas veces, en medio de una expresión matemática, abramos un paréntesis que luego se nos olvida cerrar... entonces al momento de compilar, el compilador nos indicará precisamente ese error.

- **Errores de Precaución:** son las recomendaciones para efectos de mejoramiento o aseguramiento de nuestros programas

- **Puesta en marcha:** Luego de que hemos realizado las correcciones pertinentes para que nuestro compilador nos reporte cero errores de sintaxis y cero errores de precaución ya estamos en condiciones de poner a "correr" nuestro programa o sea en condiciones de ser ejecutado por el computador.

- **Verificación de resultados**: Este último paso es útil ya que con lo que nos entregue la ejecución del programa podremos saber si se cumplió el objetivo inicial o no. En caso de que no se haya cumplido el objetivo inicial (al llegar a este punto) ser por algunas de las siguientes razones:

    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 con el que pensábamos trabajar y el programa transcrito final terminó siendo una representación técnica diferente del algoritmo inicial

  Lo que sí podemos asegurar es que, si mantenemos esta metodología paso a paso y cada uno lo realizamos concienzudamente, siempre al realizar la Verificación de Resultados se va a satisfacer con éstos el objetivo inicial.

Ejemplo 1:

ALGORTMO: Promedio final

DESCRIPCION: Elaborar un algoritmo para calcular el promedio final de la materia de algoritmos. Dicha calificación se compone de los siguientes porcentajes.

55% -----del promedio final de sus calificaciones parciales (3)

30% ----- de la calificación de promedio

15% ----- de la calificación de un trabajo final

CONSTANTE: ----------

VARIABLE: Real: P_1, P_2, P_3, Prom., Examen, TrabajoF, Prom. Final

INICIO

1. Leer P_1, P_2, P_3
2. Prom=((P_1+ P_2+ P_3)/3)*0.55
3. Leer Examen
4. Leer TrabajoF
5. Prom. Final= (Prom + (Examen*0.30) + (TrabajoF *0.15))
6. Escribir Prom. Final

FIN

Ejemplo 2

ALGORITMO: Sueldo

DESCRIPCION: Calcular el sueldo de un empleado dados como datos de entrada: el nombre, hrs. De trabajo y el pago en hr.

CONSTANTE: Real: Pagohr=50.30

VARIABLE: Cadena: nombre Entero: hrs. Real: Sueldo

INICIO

1. Leer nombre
2. Leer hrs.
3. Sueldo= Pagohr*hrs
4. Escribir Sueldo, nombre

FIN

Ejemplo 3

ALGORITMO: Evaluación

DESCRIPCION: Elaborar un algoritmo que obtenga e imprima el valor de Y a partir de la ecuación.

$Y= 3*X^2 + 7X - 15$

CONSTANTE: ----------

VARIABLE: Real: X, Y

INICIO

1. Leer X
2. Y= (3*X*X)+(7*x) -15
3. Escribir Y

FIN

Ejemplo 4

ALGORITMO: Distancia recorrida por un móvil

DESCRIPCIÓN: Se desea calcular la distancia recorrida (m) por un móvil que tiene velocidad constante (m/s), durante un tiempo T (Sg), considerar que es un MRU (Movimiento Rectilíneo Uniforme).

VARABLES: Real: d,t,v

INICIO

1. Leer v
2. Leer t
3. D=v*t
4. Escribir d

FIN

BIBLIOGRAFÍA

Wikipedia. (2019). Definición de indentación. Obtenido: https://es.wikipedia.org/wiki/Indentación

Burítica, O. I. (1999). *La esencia de la lógica de la programación .* Pereira: Papiro.

Joyanes, L. (2008). Fundamentos de programación: Algoritmos, estructura de datos y objetos, 4ta Edición . Madrid: McGraw-Hill.