Capítulo 2

Metodología para Solucionar un Problema

Siempre que vamos a resolver un problema nos enfrentamos con la dificultad de tener que encontrar precisamente eso: Una Solución. Pocas veces nos detenemos a pensar que existe un camino estructural que nos permite resolver cualquier problema (en términos generales) teniendo, como es obvio, que entrar en la minucia del detalle dependiendo del problema.

Cuál es el primer paso que debemos dar cuando nos enfrentamos a un problema...? Lo primero que debemos tener muy pero muy muy claro es Cual es el problema. 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. De manera que lo primero a conocer muy bien es el problema como tal que en nuestros términos lo vamos a ver no como un problema sino como un **objetivo**, lo voy a volver a decir, no lo vamos a ver como un problema sino como un

Objetivo

Sí, es lo mas importante en el desarrollo de un problema y por ello he agrandado la letra en esta palabra solo para que usted sepa que es por ahí por donde debe comenzar. Tener claro el objetivo nos va a permitir obtener dos beneficios que a la postre serán mas grandes de lo que podemos pensar:

a. Tener claro el objetivo nos permite saber hacia donde vamos
b. Tener claro el objetivo nos permite saber hasta donde debemos llegar

Estas dos definiciones parecieran ser lo mismo pero en el fondo no lo son. Usted puede tener muy claro hacia donde va pero puede no saber hasta donde debe llegar o, dicho en otras palabras, no saber en donde debe parar ó podría saber en donde debe para pero no tener ni idea por cual ruta llegar. El objetivo se ha de convertir en la razón de ser en la solución de un problema.

Alguna vez, antes de irme a estudiar a la ciudad de Bogotá, mi padre en una de esas tardes en las cuales se sentaba a aconsejarme me dijo Te vas a ir a Bogotá con el objetivo de estudiar. Vas a tener toda la libertad del mundo para hacer todo lo que querás pero eso sí, independiente de todo lo que hagás en la capital metete en tu cabeza que la clave del éxito para cualquier cosa en la vida es No Perder de Vista el Objetivo cualquiera que este sea. Desde allí entendí que realmente tener un objetivo claro, verdaderamente claro, exageradamente claro es mas importante que cualquier otra cosa, porque gracias a ello puede uno ir detrás de dicho objetivo hasta lograrlo.

En nuestro caso, y mas que nunca, podemos decir que para llegar a la solución de un problema la clave de ello está en **Tener muy claro cuál es el objetivo** y **No perderlo nunca de Vista**. Tal vez usted tendrá alguna inquietud en cuanto a la insistencia de este tópico pero la realidad es que muchas veces creemos tener claro el objetivo y solo cuando nos empeñamos en lograrlo vemos que no era así.

Tener claro el objetivo nos permite algo adicional. Aquí voy a utilizar una frase que, aunque un poco romántica, nos va a ilustrar a que me refiero: El objetivo es el faro que solo cuando está bien claro nos ilumina el camino para lograrlo. Cuando el objetivo está suficientemente claro podemos vislumbrar un camino lógico para llegar hasta él. Ese camino lógico va a tener un nombre dado la orientación de este libro y ese nombre es

Algoritmo

Qué es un Algoritmo...? Es un conjunto de pasos secuenciales y ordenados que permiten lograr un objetivo. Que sean pasos secuenciales significa que deben ser ejecutados uno después de otro y que sean pasos ordenados quiere decir que deben llevar un orden quasi-obligatorio (u obligatorio en la mayoría de los casos). Como puede notar el algoritmo permite lograr un objetivo. O sea que éste es el camino que necesitamos para lograrlo.

De nuevo, cuándo podemos vislumbrar claramente el algoritmo..? Cuando el objetivo está realmente claro. Siempre que usted, en el desarrollo de la solución de un problema, vea que en algún momento no sabe por donde coger, no sabe qué hacer o se siente perdido, no busque mas simplemente quiere decir que realmente usted no tenía tan claro el objetivo como había pensado.

Como se estructura un objetivo..? Muy sencillo, esto le va a parecer muy obvio pero aún así se lo voy a decir. Un algoritmo se estructura comenzando en un Inicio y terminando en un Fin. Mucho de

lo que encuentre en este libro notará que es exageradamente lógico pero no olvide no solo el título de este libro sino también la primera parte del primer capítulo del mismo.

Veamos entonces ejemplo: Desarrollar un algoritmo que nos permita adquirir el libro **El Coronel no tiene quien le escriba** de Gabriel García Márquez.

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.

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.

Explicado así el algoritmo no va a pasar de ser un breve texto explicativo que nos va a permitir lograr algo y que en este caso es la adquisición de un Libro determinado. Pero podríamos organizar este algoritmo de manera que fuera un poco mas estético y, porqué no decirlo, un poco mas entendible comenzando por el hecho de que esta vez le vamos a colocar un nombre al algoritmo y que lo vamos a generalizar para conseguir cualquier libro siempre y cuando esté completamente definido

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

Note algunas puntualizaciones al respecto de este algoritmo:

a. Casi todas las líneas van numeradas, pero no todas.
b. En la línea 1 se debe cumplir esa orden para poder continuar con el resto del algoritmo, porque se asume en algoritmo que no solo se pasa por encima de las líneas sino que se realizan las tareas allí indicadas.
c. 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.
d. 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.
e. 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...

Puede notarse en este tipo de algoritmos dos cosas: Primero que son algoritmos conocidos como informales, es decir aquellos algoritmos (según los libros) que no pueden ser implementados a través de un computador. Yo sería un poco menos drástico. Yo diría que son algoritmos informales aquellos que no son fácilmente implementables en un computador. Segundo y precisamente debido a que son algoritmos informales, deben hacerse una cantidad de reflexiones antes y después de ellos. Reflexiones que tienen una connotación puramente humana.

En la reflexión e. se habló de una prueba. Textualmente dice “Si probamos este ejemplo ....”. Ello significa que todo algoritmo debe ser probado antes de ser ejecutado con el propósito de que tengamos una alta certeza en cuanto al logro del objetivo. Precisamente éste es el tercer concepto que tendremos claro

La prueba

Que para efectos técnicos se llamará la Prueba de Escritorio. Qué es pues la Prueba de Escritorio..? Es la simulación de la puesta en marcha de un algoritmo. Con la Prueba de Escritorio podemos determinar si el algoritmo que hemos diseñado logra el objetivo propuesto. De no ser así podremos concluir que se debe corregir el algoritmo hasta lograr que satisfaga el objetivo propuesto.

Por lo que usted ha podido observar en el algoritmo de ejemplo, cada línea numerada del algoritmo puede considerarse a su vez otro algoritmo ya que el solo hecho de Saber cuál es el libro que se quiere adquirir nos obliga a realizar una serie de pasos ordenados y secuenciales para poderlo saber. Entonces surge una inquietud... Qué tan detallado puede ser un algoritmo..? Su respuesta, como todo lo que va a encontrar en este libro es muy lógica y muy sencilla. Un algoritmo debe tener el nivel de detalle suficiente como para que no exista ninguna duda en su puesta en marcha, es decir, como para que cada línea pueda ser realizada sin el mas mínimo asomo de inquietud. Ello quiere decir que algunos algoritmos pueden ser más entendibles para unas personas que para otras dada su misma definición racional.

Como todo dentro del conocimiento humano requiera una clasificación y los conceptos de los cuales estamos hablando no son la excepción, los algoritmos se clasifican en :

Algoritmos Informales

Definidos como todos aquellos algoritmos que no son realizables a través de un computador o al menos no fácilmente. Son aquellos algoritmos en donde el ejecutor real es el ser humano como el

algoritmo para dar un beso, el algoritmo para fritar unos huevos o el algoritmo para conseguir un libro. Escribo que al menos no fácilmente porque la tecnología ha avanzado tanto que muchos algoritmos que en el pasado no eran implementables a través de un computador en la actualidad lo son y de manera mucho más sencilla como es el caso del algoritmo para conseguir un libro que anteriormente se pensaba en librerías y ahora se piensa en un concepto mas globalizado: Internet, con mas posibilidad de conseguirlo y con menos trabajo.

De manera que vamos a considerar aquellos algoritmos informales como los que son preferiblemente realizables por el ser humano.

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 si 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 mas rápido y absolutamente confiable (de hecho depende de que el algoritmo igualmente sea muy confiable). Son precisamente estos algoritmos los que vamos a tratar de definir y poner en práctica en el desarrollo de este libro.

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:

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.

Qué son las reglas sintácticas de un Lenguaje de Programación..? Son todas las restricciones técnicas (y algunas veces caprichosas) sobre las cuales está construído el Lenguaje. Por ejemplo y solo con el ánimo de ilustrar lo que acabo de decir. Si estamos utilizando el Lenguaje de Programación C, la orden para leer un dato se da con la instrucción cin así como está escrito y sin ningún tipo de modificación por mas mínima que ésta sea. Será un error, entonces, escribir esta orden así Cin ó así cim. El Lenguaje C solo entiende la instrucción cin tal como sus creadores la diseñaron. De tal forma que para escribir un algoritmo computacional en términos entendibles a un computador lo único que necesitamos saber son las reglas sintácticas de un Lenguaje de Programación cualquiera. El algoritmo escrito con dichas reglas se llamará Programa.

Qué es pues un Programa..? Es un algoritmo escrito con las instrucciones, las restricciones y las reglas de un Lenguaje de Programación.

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 mas que un texto escrito bajo la óptica de algunas reglas preestablecidas por los creadores de un Lenguaje de Programación.

Compilación

Es muy normal que al reescribir un algoritmo con las reglas sintácticas de un Lenguaje de Programación es decir al escribir un programa, omitamos algunas reglas y se nos vayan, sin querer, algunos errores. Por ejemplo que en alguna parte del programa abrimos un paréntesis que luego se nos olvidó cerrar. Para ello el computador nos facilita una herramienta que revisa la sintaxis del programa, nos dice si tiene errores y, en los casos mas depurados, nos dice en qué líneas del programa están los errores y hasta nos sugiere la corrección.

Entonces qué es la 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. 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.- Podríamos asociar los errores de sintaxis en un Lenguaje de Programación con los errores de Ortografía en nuestro idioma. Son aquellos errores representados en la omisión de alguna o algunas reglas sintáctica (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.- Algunos compiladores nos hacen, por decirlo así, cierto tipo de recomendaciones para efectos de mejoramiento o aseguramiento de nuestros programas. Este tópico lo veremos de manera mas detallada en la medida que se vayan desarrollando los temas de este libro.

Porqué se habla de algunos compiladores..? Pues porque dado que existen varios Lenguajes de Programación, cada Lenguaje de Programación tiene su propio compilador o sea su propio revisor sintáctico. Podríamos decir de nuevo (y aunque sea mal dicho sirve en este caso) que la sintaxis es a un Lenguaje de Programación como la ortografía es a un Idioma.

Porqué existen varios Lenguajes de Programación..? Esto sí obedece a dos factores: el primero es la especificidad de los Lenguajes ya que son desarrollados para que cumplan de la mejor manera ciertos objetivos (refiriéndonos al mundo de la informática) y el segundo es un factor netamente comercial pues los Lenguajes de Programación son producidos por empresas fabricantes de software.

En un programa los errores son de tres tipos: Errores de Sintaxis y Errores de Precaución que como ya se dijo son revisados por el compilador. Son los errores fáciles porque los compiladores actuales no solo le dicen a uno cuál es el error sino que además le indican mas o menos en donde está e incluso algunas veces le sugieren la corrección. Los errores difíciles realmente de encontrar

en un programa son el tercer tipo de error y son los Errores Lógicos ya que el compilador no le va a discutir acerca de lo que usted quiere hacer y cómo quiere hacerlo.

Y en donde se detectan los Errores Lógicos..? Pues en la Prueba de Escritorio, allí y solo allí usted podrá determinar si su algoritmo está realmente bien o no es decir si logra o no el objetivo propuesto. Ha de tenerse especial cuidado cuando un algoritmo sea transcrito ya que el cambio de cada línea de un algoritmo por su correspondiente instrucción en un programa a veces cambia un poco la lógica inicial, si no se conocen bien las reglas sintácticas del Lenguaje de Programación.

Ejecución o 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. Si lo que queríamos inicialmente (o sea nuestro objetivo) era generar los 100 primeros números pares entonces al momento de la ejecución deberán aparecer en pantalla los 100 primeros números pares.

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.

Ejercicios sobre Algoritmos Informales

La única forma como uno puede realmente aprender a nadar o a tirarse desde un paracaídas es haciéndolo por eso lo invito a que se siente pacientemente a desarrollar estos algoritmos pensados

para que usted encuentre una gran coincidencia entre unos y otros a pesar de tener objetivos diferentes. Sé que surgirán muchas dudas en cuanto a algunos de ellos pero también estoy seguro que si usted lee este libro detenidamente va a despejar todas las dudas que tenga de tal forma que tome al azar cualquiera de los siguientes enunciados y siéntese a practicar y poner a funcionar un poquito esa lógica humana que tan pocas veces ejercitamos. (algunas posibles soluciones se encuentran en el Libro *Algoritmos* del mismo autor).

1. Desarrollar un algoritmo que permita adquirir una revista.
2. Desarrollar un algoritmo que permita entrar a una casa que está con llave.
3. Desarrollar un algoritmo que permita dar un beso.
4. Desarrollar un algoritmo que permita empacar un regalo.
5. Desarrollar un algoritmo que permita encender un vehículo.
6. Desarrollar un algoritmo que permita fritar un huevo.
7. Desarrollar un algoritmo que permita mirar por un telescopio.
8. Desarrollar un algoritmo que permita botar la basura.
9. Desarrollar un algoritmo que permita tomar un baño.
10. Desarrollar un algoritmo que permita estudiar para un examen.
11. Desarrollar un algoritmo que permita tocar determinada canción con un instrumento musical.
12. Desarrollar un algoritmo que permita viajar en avión.
13. Desarrollar un algoritmo que permita encender un bombillo.
14. Desarrollar un algoritmo que permita encender una vela.
15. Desarrollar un algoritmo que permita apagar una vela.
16. Desarrollar un algoritmo que permita apagar un bombillo.
17. Desarrollar un algoritmo que permita parquear un vehículo.
18. Desarrollar un algoritmo que permita almorzar.
19. Desarrollar un algoritmo que permita ir de la casa al trabajo.
20. Desarrollar un algoritmo que permita colocarse una camisa.
21. Desarrollar un algoritmo que permita quitarse la camisa.
22. Desarrollar un algoritmo que permita escuchar un determinado disco.
23. Desarrollar un algoritmo que permita abrir una ventana.
24. Desarrollar un algoritmo que permita ir a la tienda a comprar algo.

25. Desarrollar un algoritmo que permita tomar una fotografía.
26. Desarrollar un algoritmo que permita hacer deporte.
27. Desarrollar un algoritmo que permita cortarse el cabello.
28. Desarrollar un algoritmo que permita hacer un avión con una hoja de papel.
29. Desarrollar un algoritmo que permita manejar una bicicleta.
30. Desarrollar un algoritmo que permita manejar una motocicleta.
31. Desarrollar un algoritmo que permita manejar un monociclo.
32. Desarrollar un algoritmo que permita maquillarse.
33. Desarrollar un algoritmo que permita hacer un pastel.
34. Desarrollar un algoritmo que permita hacer un almuerzo.
35. Desarrollar un algoritmo que permita adquirir un pantalón.
36. Desarrollar un algoritmo que permita hacer un mercado pequeño.
37. Desarrollar un algoritmo que permita leer el periódico.
38. Desarrollar un algoritmo que permita saludar a un amigo.
39. Desarrollar un algoritmo que permita arrullar a un bebé hasta que se duerma.
40. Desarrollar un algoritmo que permita hacer un gol en fútbol.
41. Desarrollar un algoritmo que permita jugar ping-pong.
42. Desarrollar un algoritmo que permita nadar.
43. Desarrollar un algoritmo que permita tirarse desde un avión con un paracaídas.
44. Desarrollar un algoritmo que permita tirarse desde un avión sin un paracaídas.
45. Desarrollar un algoritmo que permita descifrar un jeroglífico.
46. Desarrollar un algoritmo que permita amarrase un zapato.
47. Desarrollar un algoritmo que permita quitarse los zapatos.
48. Desarrollar un algoritmo que permita silbar.
49. Desarrollar un algoritmo que permita elevar una cometa.
50. Desarrollar un algoritmo que permita desarrollar algoritmos.