**TEMA 1**

**ALGORITMOS[[1]](#footnote-1)**

*Ing. Carlos Balderrama Vásquez.*

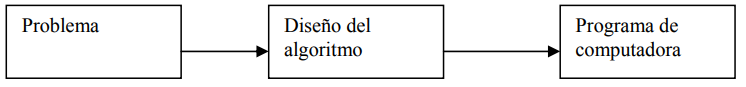
**1.1 Introducción**

El objetivo principal de este material es el de apoyar al estudiante para resolver problemas mediante una computadora. Un programador, antes que nada, es un solucionador de problemas.

Por lo que para llegar a ser un programador eficaz es necesario aprender a resolver problemas de un modo riguroso y sistemático.

Antes de comenzar con el tema vamos a considerar el significado de la palabra ***ALGORITMO*** esta palabra se deriva de la traducción al latín de la palabra árabe ***ALKHÔWARÎZMI***, nombre de un matemático y astrónomo árabe que escribió un tratado sobre la manipulación de números y ecuaciones en el siglo IX, titulado ***KITAB AL-JABR W’ALMUGALABA***, la palabra álgebra se derivó por su semejanza sonora de ***AL-JABR***.

Etimológicamente la palabra problema deriva del griego ***PROBALLEIN*** y significa “***algo lanzado hacia delante***”. Un problema es un asunto o conjunto de cuestiones que se plantean para ser resueltas, la naturaleza de los problemas varia con el ámbito o con el contexto donde están planteados: así existen problemas matemáticos, físicos, filosóficos, etc.,



**1.2 Definición**

“Un Algoritmo es una secuencia de operaciones detalladas y no ambiguas, que al ejecutarse paso a paso, conducen a la solución de un problema”. En otras palabras es un conjunto de reglas para resolver una cierta clase de problema.

“Algoritmo es un conjunto de instrucciones que especifican la secuencia de operaciones a realizar, en orden, para resolver un sistema específico o clase de problema”.

“Un Algoritmo es la aplicación de pasos lógicos, secuenciales y metódicamente aplicados para dar solución a un problema en cuestión.” En otras palabras, un algoritmo es una fórmula para resolver problemas.

“Todo problema se puede describir por medio de un algoritmo “

**1.3 Características.**

Las propiedades de un algoritmo son las siguientes:

a) El algoritmo debe ser **preciso** e **indicar el orden** de realización de cada paso.

b) El algoritmo debe ser **definido**, si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.

c) El algoritmo debe ser **finito**, si se sigue un algoritmo se debe terminar en algún momento; o sea debe tener un número finito de pasos.

**1.4 Clasificación.**

Se puede clasificar de la siguiente forma:

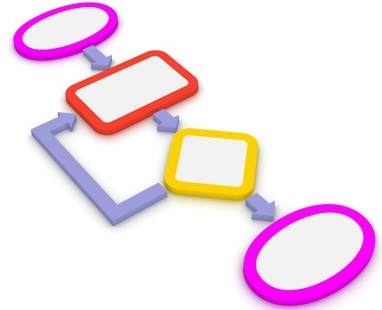
• Secuenciales

• Condicionales

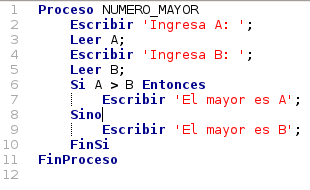
• Repetitivos

**1.5 Tipos de Lenguajes Algorítmicos**

• Gráficos: Es la representación gráfica de las operaciones que realiza un algoritmo (diagrama de flujo).



• No Gráficos: Representa en forma descriptiva las operaciones que debe realizar un algoritmo (pseudocódigo).



Un algoritmo puede ser expresado de las siguientes formas:

a) Lenguaje Natural: el uso de términos del lenguaje natural, es una forma de representar un algoritmo.

b) Lenguaje Simbólico: es otra forma de representación de un algoritmo, que además permite una introducción a la programación estructural.

c) Lenguaje Gráfico: es una forma de escribir una secuencia de pasos en forma de diagrama, en la práctica se denomina Diagramas de Flujo.

Una analogía de lo anterior es: Una receta de un plato de cocina se puede expresar en español, inglés o francés, pero cualquiera que sea el lenguaje, los pasos para la elaboración del plato se realizarán de la misma forma sin importar el cocinero.

**1.6 Algunas técnicas para el diseño de Algoritmos.**

El diseño de la mayoría de los algoritmos requiere creatividad y conocimientos de la metodología de programación, esto significa que debe desarrollar una lógica computacional que pueda resolver el problema en cuestión de una manera sencilla y óptima con las estructuras de control adecuadas.

En el diseño de un algoritmo, debemos partir del análisis del problema. Con esto podemos decir que esta tarea, difícilmente podría llegar a automatizarse en sistemas de cómputo, de tal forma que idear un algoritmo continua siendo una labor bastante creativa donde los conocimientos y las experiencias del propio diseñador tiene un papel fundamental.

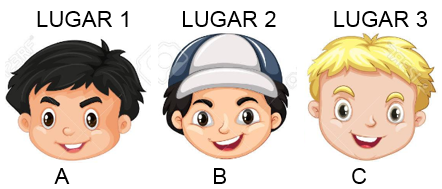
**1.6.1 Divide y vencerás.**

Esta técnica consiste en descomponer un problema en subproblemas, resolver independientemente estos subproblemas para luego combinar sus soluciones y obtener la solución completa del original.

Como ejemplo, esta técnica es utilizada con éxito en problemas matemáticos como la multiplicación de matrices, ordenación de vectores, etcétera.

**1.6.1.1Ejemplo:**

¿De cuántas maneras diferentes se pueden sentar 3 niños en una fila seguidos uno de otro?



Para solucionar este problema lo vamos a dividir:

a) Primero tenemos que determinar ¿de cuántas maneras se puede llegar a sentar el niño “A” en los tres lugares diferentes y desocupados a los que llega? Revisando la única posibilidad la respuesta es 3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **POSIBILIDAD** | **Lugar 1** | **Lugar 2** | **Lugar 3** | **Formas en que “A” puede ocupar un lugar en la fila** |
| **1** | A | A | A | **3** |

**(Nota**: *Marcaremos en color Azul los lugares disponibles que se podrían ocupar y en color Rojo los lugares ya ocupados.*

De esta forma hemos solucionamos nuestro primer problema.

b) A continuación habrá que determinar ¿de cuántas maneras diferentes se puede llegar a sentar el segundo niño “B” teniendo en cuenta que el niño “A” ya ocupa un lugar. Revisando las tres posibilidades en las que esto sucede, la respuesta es 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **POSIBILIDAD** | **Lugar 1** | **Lugar 2** | **Lugar 3** | **Formas en que “B” puede ocupar un lugar en la fila** |
| **1** | A | B | B | **2** |
| **2** | B | A | B | **2** |
| **3** | B | B | A | **2** |

c) Acto seguido con el escenario anterior habrá que determinar ¿de cuántas maneras diferentes se puede llegar a sentar el tercer niño “C” teniendo en cuenta que el niño “A” y el “B” ya ocupan un lugar. Revisando las seis posibilidades en las que esto sucede, la respuesta es 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **POSIBILIDAD** | **Lugar 1** | **Lugar 2** | **Lugar 3** | **Formas en que “C” puede ocupar un lugar en la fila** |
| **1** | C | A | B | **1** |
| **2** | C | B | A | **1** |
| **3** | A | C | B | **1** |
| **4** | B | C | A | **1** |
| **5** | A | B | C | **1** |
| **6** | B | A | C | **1** |

De esta forma hemos llegado a la respuesta del problema inicial: ¿De cuántas maneras diferentes se pueden sentar 3 niños en una fila seguidos uno de otro?

Respuesta: 6.

**1.6.2 Método voraz.**

Este método trata de producir un tipo de mejor resultado a partir de un conjunto de opciones candidatas. Para ello se va procediendo paso a paso realizándose la mejor elección de entre las posibles. Puede emplearse en problemas de optimización o la planificación en la ejecución de programas en una computadora, etcétera.

**1.6.2.1 Ejemplo:** Un cliente ha comprado un artículo con valor de $250 y nos ha pagado con un billete de $10,000



Las opciones de cambio son:

Billetes de: $500, $200, $100, $50 y 20.

**Problema**: Se necesita dar el cambio, utilizando el menor número de billetes.

**Solución**: La solución es dividir la cantidad de cambio a regresar (en este caso son $9,750) consecutivamente desde los billetes de más alta denominación, hasta los de más baja.

a) 9,750/500 = 19.5 tenemos que entregar **19 billetes de 500** y el residuo es **250**

b) 250/200 = 1.25 tenemos que entregar **1 billete de 200** y el residuo es **50**

c) 50/50 = 1 tenemos que entregar **1 billete de 50** y el residuo es **0**

Por lo tanto, la respuesta es 21 billetes.

El estilo y calidad de los algoritmos van fuertemente unidos ante la pregunta ¿cuáles son las características de un buen algoritmo? La respuesta identifica los factores de calidad de los algoritmos.

• Corrección: el algoritmo debe funcionar.

• Eficiencia : el algoritmo no debe desaprovechar recursos.

• Claridad : el algoritmo debe estar bien documentado.

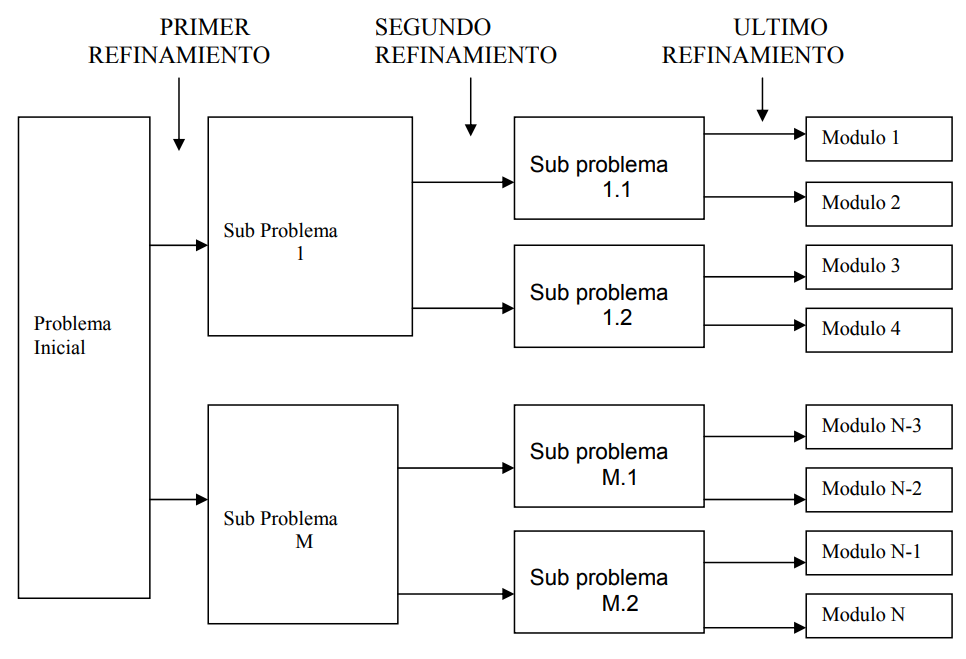
**1.6.3 Top Down**

También conocida como de arriba-abajo (diseño descendente) y consiste en establecer una serie de niveles de mayor a menor complejidad (arriba-abajo) que den solución al problema. Consiste en efectuar una relación entre las etapas de la estructuración de forma que una etapa jerárquica y su inmediato inferior se relacionen mediante entradas y salidas de información.

Este diseño consiste en una serie de descomposiciones sucesivas del problema inicial, que recibe el refinamiento progresivo del repertorio de instrucciones que van a formar parte del programa.

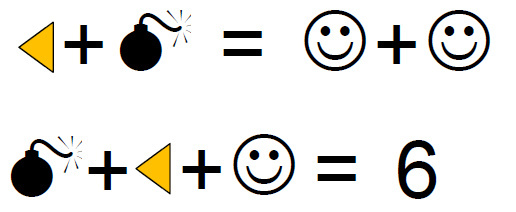
La utilización de la técnica de diseño Top-Down tiene los siguientes objetivos básicos:

* Simplificación del problema y de los subprogramas de cada descomposición.
* Las diferentes partes del problema pueden ser programadas de modo independiente e incluso por diferentes personas.
* El programa final queda estructurado en forma de bloque o módulos lo que hace mas sencilla su lectura y mantenimiento.



**1.6.3.1 Ejemplo:**

* ¿Cuánto vale la carita blanca?[[2]](#footnote-2)



Para poder solucionar el problema vamos realizando el análisis TOP DOWN.

De la primera parte no podemos hacer una inferencia directa ya que no tenemos elementos suficientes. Entonces podemos echar mano de la segunda parte donde observamos que las tres figuras diferentes (la bomba, el triángulo y la carita blanca) suman 6.

Por la propiedad conmutativa de la suma la bomba y el triángulo (en cualquier orden de su suma) son equivalentes a la suma de dos caritas blancas.

Esto implica que la suma de 3 caritas blancas es seis. De aquí, deducimos que el valor de cada una de las caritas blancas es de 2.

**1.6.4 Bottom Up**

En este diseño, las partes individuales se diseñan con detalle y luego se enlazan para formar componentes más grandes, que a su vez se enlazan hasta que se forma el sistema completo. Las estrategias basadas en el flujo de información "bottom-up" se antojan potencialmente necesarias y suficientes porque se basan en el conocimiento de todas las variables que pueden afectar los elementos del sistema.

1. *Documento tomado del sitio:* [*http://moodle2.unid.edu.mx/dts\_cursos\_mdl/lic/IC/EA/AM/06/Algoritmos.pdf*](http://moodle2.unid.edu.mx/dts_cursos_mdl/lic/IC/EA/AM/06/Algoritmos.pdf) *y modificado para fines educativos por Héctor Rasso Mora. Colegio de Bachilleres México. Febrero 2018* [↑](#footnote-ref-1)
2. Zuñiga J.A. y Zúñiga E. (2007). *Introducción a la resolución de problemas*. México D.F.: Progreso. pp 85 [↑](#footnote-ref-2)