ALGORITMOS: IntroducciÃ³n

p@ul

- **Etapas en la ResoluciÃ³n de Problemas por Computador**

La resoluciÃ³n de problemas utilizando como herramienta una computadora no se resume Ãºnicamente en la escritura de un programa, sino que se trata de una tarea mÃ¡s compleja. El proceso abarca todos los aspectos que van desde interpretar las necesidades del usuario hasta verificar que la respuesta brindada es correcta. Las etapas son las siguientes:

AnÃ¡lisis del problema:

En esta etapa, se analiza el problema en su contexto del mundo real. Deben obtenerse los requerimientos del usuario. El resultado de este anÃ¡lisis es un modelo preciso del ambiente del problema y del objetivo a resolver. Dos componentes importantes de este modelo son los datos a utilizar y las transformaciones de los mismos que llevan al objetivo.

DiseÃ±o de una soluciÃ³n:

La resoluciÃ³n de un problema suele ser una tarea muy compleja para ser analizada como un todo. Este aspecto puede facilitarse mediante la identificaciÃ³n de las partes (subproblemas) que componen el problema y la manera en que se relacionan. Cada uno de estos subproblemas debe tener un objetivo especÃ–fico, es decir, debe resolver una parte del problema original. La uniÃ³n de todos los subproblemas es lo que permitirÃ¡ obtener la soluciÃ³n buscada.

EspecificaciÃ³n de algoritmos:

Cada uno de los subproblemas que componen la soluciÃ³n deben ser especificados a travÃ©s de un algoritmo. Esta etapa busca obtener la secuencia de pasos a seguir para resolver el problema. La elecciÃ³n del algoritmo adecuado es fundamental para garantizar la eficiencia de la soluciÃ³n.

Escritura de programas:

Un algoritmo es una especificaciÃ³n simbÃ³lica que debe convertirse en un programa real sobre un lenguaje

de programaciÃ³n concreto. A su vez, un programa escrito en un lenguaje de programaciÃ³n determinado (ej: Pascal, Ada, etc) es traducido automÃ¡ticamente al lenguaje de mÃ¡quina de la computadora que lo va a ejecutar. Esta traducciÃ³n, denominada compilaciÃ³n, permite detectar y corregir los errores sintÃ¡cticos que se cometan en la escritura del programa.

VerificaciÃ³n:

Una vez que se tiene un programa escrito en un lenguaje real se debe verificar que su ejecuciÃ³n produce al resultado deseado, utilizando datos representativos del problema real. SerÃ–a deseable poder afirmar que el programa es correcto, mÃ¡s allÃ¡ de los datos particulares de una ejecuciÃ³n. Sin embargo, en los casos reales es muy difÃ–cil realizar una verificaciÃ³n exhaustiva de todas las posibles condiciones de ejecuciÃ³n de un sistema de software. La facilidad de verificaciÃ³n y la depuraciÃ³n de errores de funcionamiento del programa conducen a una mejor calidad del sistema y es un objetivo central de la IngenierÃ–a de Software.

Los programas deben estar escritos en un lenguaje comprensible por la mÃ¡quina, ya sea:

- Lenguaje mÃ¡quina
- CÃ³digo comprensible por un intÃ©rprete/compilador (que traduce el programa a la mÃ¡quina).

Lenguaje mÃ¡quina: Para obtener el cÃ³digo que comprende la mÃ¡quina (ceros y unos) es necesario someter al cÃ³digo fuente, cÃ³digo comprensible por el ser humano, a un proceso de traducciÃ³n:

1. Compilar: Realizar AnÃ¡lisis LÃ©xico, SintÃ¡ctico y SemÃ¡ntico, ConversiÃ³n a cÃ³digo objeto.

2. Enlazado (*linkado*): Unir el cÃ³digo objeto a las librerÃ–as y datos necesarios para que el sistema operativo sepa ejecutar el programa y convertirlo en un ejecutable.

Ejemplo de lenguajes habitualmente *compilables*: Pascal, C, C++, Ada.

CÃ³digo interpretable: El cÃ³digo comprensible por el ser humano es convertido en tiempo de ejecuciÃ³n (equivalente a una traducciÃ³n simultÃ¡nea) a cÃ³digo comprensible por la mÃ¡quina.

Se diferencia del cÃ³digo compilado en que:

- Es mÃ¡s lento (necesita traducirse en cada ejecuciÃ³n).
- Es mÃ¡s fÃ¡cil de mantener (modificar, corregir errores), ya que siempre es comprensible por el ser humano.
- Es mÃ¡s flexible ya que los lenguajes interpretados permiten mayores abstracciones que los lenguajes compilados.
- Es portable a cualquier plataforma que posea un intÃ©rprete. El programa compilado solo funciona en la plataforma para la que fue compilado.

Ejemplo de lenguaje habitualmente interpretado: Perl, VBS, Basic, Javascript, PHP.

Existen soluciones intermedias al cÃ³digo compilado o interpretado. Ejemplo: Java, posee *bytecodes* que es cÃ³digo compilado intermedio que requiere ser interpretado parcialmente.

- **Conceptos BÃ¡sicos de Algoritmos**

Â¿QuÃ© es un algoritmo?

- La palabra algoritmo deriva del nombre de un matemÃ¡tico Ã¡rabe del siglo IX, llamado

Al-Khuwarizmi, quien estaba interesado en resolver ciertos problemas de aritmÃ©tica y describiÃ³ varios mÃ©todos para resolverlos. Estos mÃ©todos fueron presentados como una lista de instrucciones especÃ–ficas (como una receta de cocina) y su nombre se utiliza para referirse a dichos mÃ©todos.

- Un algoritmo es, en forma intuitiva, una receta, un conjunto de instrucciones o de especificaciones sobre un proceso para hacer algo. Ese algo generalmente es la soluciÃ³n de un problema de algÃºn tipo. Se espera que un algoritmo tenga varias propiedades. La primera es que un algoritmo no debe ser ambiguo, o sea, que si se trabaja dentro de cierto marco o contexto, cada instrucciÃ³n del algoritmo debe significar sÃ³lo una cosa.
- Una posible definiciÃ³n de algoritmo es un conjunto de reglas que permiten obtener un resultado determinado a partir de ciertas reglas definidas. Otra definiciÃ³n serÃ–a, algoritmo es una secuencia finita de instrucciones, cada una de las cuales tiene un significado preciso y puede ejecutarse con una cantidad finita de esfuerzo en un tiempo finito. Ha de tener las siguientes caracterÃ–sticas: legible, correcto, modular, eficiente, estructurado, no ambiguo y a ser posible se ha de desarrollar en el menor tiempo posible.
- Un algoritmo es una secuencia no ambigua, finita y ordenada de instrucciones que han de seguirse para resolver un problema. Un programa normalmente implementa (traduce hacia un lenguaje de programaciÃ³n concreto) un algoritmo. Puede haber programas que no se ajusten a un algoritmo (pueden no terminar nunca), en cuyo caso se denomina procedimiento a tal programa. Los programas suelen subdividirse en partes menores (mÃ³dulos), de modo que la complejidad algorÃ–tmica de cada una de las partes es menor que la del programa completo, lo cual ayuda al desarrollo del programa.
- Una definiciÃ³n informal (no se considera aquÃ– una definiciÃ³n formal, aunque existe): conjunto finito de reglas que dan una secuencia de operaciones para resolver todos los problemas de un tipo dado. De forma mÃ¡s sencilla, podemos decir que un algoritmo es un conjunto de pasos que nos permite obtener un dato. AdemÃ¡s debe cumplir estas condiciones:

Â *Finitud*: El algoritmo debe acabar tras un nÃºmero finito de pasos. Es mÃ¡s, es casi fundamental que sea en un nÃºmero razonable de pasos.

Definibilidad: El algoritmo debe definirse de forma precisa para cada paso, es decir, hay que evitar toda ambigÃ¼edad al definir cada paso. Puesto que el lenguaje humano es impreciso, los algoritmos se expresan mediante un lenguaje formal, ya sea matemÃ¡tico o de programaciÃ³n para un computador.

Entrada: El algoritmo tendrÃ¡ cero o mÃ¡s entradas, es decir, cantidades dadas antes de empezar el algoritmo. Estas cantidades pertenecen ademÃ¡s a conjuntos especificados de objetos. Por ejemplo, pueden ser cadenas de caracteres, enteros, naturales, fraccionarios, etc. Se trata siempre de cantidades representativas del mundo real expresadas de tal forma que sean aptas para su interpretaciÃ³n por el computador.

Salida: El algoritmo tiene una o mÃ¡s salidas, en relaciÃ³n con las entradas.

Efectividad: Se entiende por esto que una persona sea capaz de realizar el algoritmo de modo exacto y sin ayuda de una mÃ¡quina en un lapso de tiempo finito.

A menudo los algoritmos requieren una organizaciÃ³n bastante compleja de los datos, y es por tanto necesario un estudio previo de las estructuras de datos fundamentales. Dichas estructuras pueden implementarse de diferentes maneras, y es mÃ¡s, existen algoritmos para implementar dichas estructuras. El uso de estructuras de datos adecuadas pueden hacer trivial el diseÃ±o de un algoritmo, o un algoritmo muy complejo puede usar estructuras de datos muy simples.

Aplicaciones de los Algoritmos:

A primera vista, se puede pensar que el conocimiento de estos algoritmos y estructuras de datos no tienen una aplicaciÃ³n prÃ¡ctica inmediata. Sin embargo, su conocimiento y correcta aplicaciÃ³n sirven para producir programas mejores, en el sentido de que aprovechan mejor la memoria del sistema, son mÃ¡s rÃ¡pidos, eficientes, robustos y tolerantes a fallos.

Las aplicaciones de estos algoritmos en algunos casos son inmediatas; por ejemplo, hallar el trayecto mÃ¡s corto entre dos estaciones es algo que interesa a muchos viajeros del metro y se pueden obtener aproximaciones bastante buenas del mundo real utilizando algunos de los algoritmos que obtienen distancias mÃ–nimas. Otros algoritmos sirven para procesar cadenas, lo cual sirve de base para analizadores lÃ©xicos o algoritmos criptogrÃ¡ficos, por ejemplo.

AdemÃ¡s, tener conocimientos adecuados de algoritmia y estructuras de datos facilita el poder pasar de un lenguaje de programaciÃ³n a otro con mucha mayor facilidad: puesto que ya se tiene la base, sÃ³lo hace falta superar las dificultades tÃ©cnicas particulares de cada lenguaje.

DiseÃ±ar un algoritmo para cambiar una llanta a un coche.

-

- Inicio.
- Traer gato.
- Levantar el coche con el gato.
- Aflojar tornillos de las llantas.
- Sacar los tornillos de las llantas.
- Quitar la llanta.
-

- Poner la llanta de repuesto.
- Poner los tornillos.
- Apretar los tornillos.
- Bajar el gato.
- Fin

- **Diagramas de Flujos de Datos**

    - Es la representaciÃ³n grÃ¡fica de los pasos que deben seguirse para resolver un problema.
    - El traducir una descripciÃ³n narrada a diagrama de flujo agrega claridad y precisiÃ³n a la descripciÃ³n de una tarea.
    - AdemÃ¡s, al elaborar el diagrama de flujo, se descubren situaciones que no habÃ–an sido consideradas.
    - En la elaboraciÃ³n de Ã©stos, la simbologÃ–a juega un papel muy importante, ya que debe estar adecuada a ciertos estÃ¡ndares, con el fin de que sea entendida por cualquier persona dedicada al campo de computaciÃ³n.
    - Cada una de las figuras representa una etapa en la soluciÃ³n del problema; dentro de ellas se anotan indicaciones. Las lÃ–neas seÃ±alan el flujo de la informaciÃ³n.
    - En la actualidad se emplean poco, pero resultan muy Ãºtiles cuando se comienza en el estudio de la programaciÃ³n.
    - El problema con los diagramas de flujo es que a medida que crece la complejidad de las proposiciones, tambiÃ©n crece el detalle con el que hay que dibujarlas.
    - Esto llegaba a convertirlos en figuras fraccionadas (pues de otro modo no cabrÃ–an en la hoja), y difÃ–ciles de seguir y entender.

- El equivalente simbÃ³lico de la flecha se llama goto (ir a) y estÃ¡, en tÃ©rminos generales, excluido de la programaciÃ³n moderna debido a su poder “desestructurante” y caÃ³tico sobre los programas (aunque si se usa con cuidado puede llegar a ser de alguna utilidad).
- Los sÃ−mbolos utilizados han sido normalizados por el Instituto Norteamericano de NormalizaciÃ³n (ANSI) y los mÃ¡s frecuentes utilizados son:

Reglas bÃ¡sicas para la construcciÃ³n de Diagramas de Flujo:

- Todo diagrama debe tener un inicio y fin.
- Las lÃ−neas de conexiÃ³n o de flujo deben ser siempre rectas, si es posible verticales y horizontales nunca cruzadas o inclinadas; para conseguir lo anterior es necesario apoyarse en conectores.
- Las lÃ−neas que enlazan los sÃ−mbolos entre sÃ− deben estar todas conectadas.
- Se deben dibujar todos los sÃ−mbolos de modo que se pueda seguir el proceso visualmente de arriba hacia abajo (diseÃ±o top-down) y de izquierda a derecha.
- Realizar un grÃ¡fico claro y equilibrado.
- Evitar la terminologÃ−a de un lenguaje de programaciÃ³n o mÃ¡quina.
- Utilizar comentarios al margen (si es necesario) para que Ã©ste sea entendible por cualquier persona que lo consulte.
- A cada bloque o sÃ−mbolo se accede por arriba y/o por la izquierda y se sale por abajo y/o por la derecha.
- Si el diagrama abarca mÃ¡s de una hoja es conveniente enumerarlo e identificar de donde viene y a donde se dirige.

Estructuras de Control en Diagramas de Flujo:

Estructuras Repetitivas en Diagramas de Flujo:

Ejercicio 1:Â Calcular el salario neto de un trabajador en funciÃ³n del nÃºmero de horas trabajadas, precio de la hora de trabajo y considerando unos descuentos fijos al salario bruto en concepto de impuestos (20 por 100).

Ejercicio 2: Realizar un diagrama de flujo que permita mostrar en pantalla un mensaje de mayorÃ−a o minorÃ−a de edad segÃºn sea el caso para un nombre especÃ−fico.

4. PseudocÃ³digo

- Los algoritmos se deben describir en un lenguaje que se parezca mÃ¡s al lenguaje utilizado para escribir programas de computador.
- Es decir, un lenguaje de pseudoprogramaciÃ³n, una imitaciÃ³n del cÃ³digo de las computadoras al cual se le conoce como pseudocÃ³digo.
- El pseudocÃ³digo se concibiÃ³ para superar las dos principales desventajas del diagrama de flujo:
- Es lento de crear.
- Es difÃ−cil de modificar sin un nuevo redibujo.
- Por otra parte el pseudocÃ³digo es mÃ¡s fÃ¡cil de utilizar ya que es similar al espaÃ±ol o inglÃ©s, catalÃ¡n, alemÃ¡n o francÃ©s, dependiendo del caso.
- Al contrario que los lenguajes de programaciÃ³n de alto nivel, como Pascal o Basic, no existe un conjunto de reglas que definan con precisiÃ³n lo que es y lo que no es pseudocÃ³digo.
- VarÃ−a de un programador a otro y de que tan prÃ³xima sea la descripciÃ³n al lenguaje de programaciÃ³n.
- El pseudocÃ³digo es una mezcla de lenguaje natural y sÃ−mbolos, tÃ©rminos y otras caracterÃ−sticas comÃºnmente utilizadas en uno o mÃ¡s lenguajes de alto nivel.
- TÃ−picamente se encuentran las caracterÃ−sticas en diferentes pseudocÃ³digos que se pueden

encontrar en libros de texto de programaciÃ³n.

- El pseudocÃ³digo requiere de ciertos sÃ−mbolos privilegiados que ya tienen significado preciso y establecido de antemano.
- A tales indicadores del pseudocÃ³digo se les conoce como “palabras clave”.
- Es necesario que exista una palabra clave para la seleccciÃ³n y otra para la iteraciÃ³n condicional, asÃ− como para las instrucciones adicionales y otras estructuras de control.
- Por ejemplo: La palabra “escribe” es una palabra clave que ya tiene significado predefinido, a diferencia de la palabra ALFA, que es una variable libre.
- Se pretenderÃ¡ uniformizar el pseudocÃ³digo utilizando la siguiente simbologÃ−a:

- El algoritmo comienza con la palabra Inicio y termina con la palabra Fin. Entre estas palabras, es escribe una instrucciÃ³n (acciÃ³n) por lÃ−nea o se separan con un punto y coma.
- La lÃ−nea encerrada entre llaves ({…}) se denomina comentario (es una informaciÃ³n al lector del programa y no realiza ninguna instrucciÃ³n ejecutable, sÃ³lo tiene efectos de documentaciÃ³n interna del programa.
- La asignaciÃ³n se llevarÃ¡ a cabo mediante el signo . Ejemplo: A 10, a la variable A se le asigna el valor de 10.
- Por lo tanto, el PseudocÃ³digo a utilizar incluirÃ¡:

    - ♦ Nombre del Programa.
    - ♦ SecciÃ³n de Declaraciones (Variables y Constantes)
    - ♦ Algoritmo.

Ejemplo: Elaborar un programa que calcule la sumatoria de 2 nÃºmeros.

5. Ejercicios

Se desea obtener una Tabla con las depreciaciones acumuladas y los valores reales de cada aÃ±o de un automÃ³vil comprado en $ 1,800 en el aÃ±o 1992, durante los seis aÃ±os siguientes; asumiendo un valor de recuperaciÃ³n de $ 120. Realizar el anÃ¡lisis del problema, conociendo la fÃ³rmula de la depreciaciÃ³n anual constante D para cada aÃ±o de vida Ãºtil.

Ejercicios Propuestos:

- Calcular el Ã¡rea de un triÃ¡ngulo.
- Calcular el precio de un artÃ−culo tras aplicarle un 18% de IGV.
- Intercambiar el contenido de dos variables.
- Resolver una ecuaciÃ³n de segundo grado, teniendo en cuenta todas las posibles alternativas.
- Se pide determinar el mayor de tres nÃºmeros ingresados por el usuario. Considere que pueden ser iguales.
- Se pide determinar el menor de tres nÃºmeros ingresados por el usuario. Considere que pueden ser iguales.
- Calcular el exponencial de un nÃºmero (ab), considerando todos los casos posibles:

    - Ingreso de nÃºmeros negativos.
    - Ingreso de valores igual a 0 (a=0, b=0).

- Calcule e imprima los primeros N nÃºmeros primos. Considere que N es ingresado por el
- usuario.
- Calcule el factorial de un nÃºmero ingresado por el usuario.

Trabajo Encargado:

- Enumerar las diferencias entre un lenguaje compilado y un lenguaje interpretado.

- Clasificar entre lenguaje compilado y lenguaje interpretado (emplear Internet para buscar informaciÃ³n en caso de ser necesario). Justifica tu respuesta.

    - Pascal
    - C
    - ADA
    - COBOL
    - Python
    - BASH
    - Basic
    - Java

- DiseÃ±ar un diagrama de flujo para “El desarrollo de un programa”.

- Escribir el pseudocÃ³digo y el diagrama de flujo para resolver los siguientes problemas:

    - Un video club necesita un programa para gestionar los alquileres. El programa consta de diferentes mÃ³dulos y se nos ha encargado diseÃ±ar el referente a la introducciÃ³n de nuevos clientes.

    - En un proceso industrial es necesario un sistema tolerante a fallos. El proceso consiste en llenar botellas de un refresco mediante el uso de un robot que emplea una pipeta. Es posible que el robot falle y la pipeta no se introduzca adecuadamente, con lo que hay que repetir la operaciÃ³n de introducir la pipeta. En determinados casos la pipeta se rompe y hay que detener el proceso y llamar al servicio tÃ©cnico.

    - Un cliente desea un sistema de mensajerÃ–a para sus empleados en las distintas sedes alrededor del mundo. En el momento que un empleado recibe un mensaje, Ã©ste es almacenado hasta que el destinatario consulta su buzÃ³n. En determinados casos el mensaje es urgente y debe enviarse un mensaje SMS al mÃ³vil del empleado. A veces un mensaje se extravÃ–a (el nombre del destinatario no coincide con ninguno almacenado en la base de datos), en esos casos es necesario enviar el mensaje de vuelta al remitente. Describir el proceso a seguir para tratar los mensajes.

    - Describir el proceso a seguir para descargar una pelÃ–cula empleando edonkey. Tener en cuenta que no disponemos del programa (pero sÃ– de conexiÃ³n a internet), que es posible que la pelÃ–cula no estÃ© disponible en el primer servidor al que conectamos e incluso que la pelÃ–cula no estÃ© disponible en ningÃºn servidor (se advierte del problema y el proceso falla).