TALLER DE ALGORITMOS Y FUNDAMENTOS DE PROGRAMACION

Sarah Valentina Sánchez Torres

Ilse Dayana Alfonso Bonilla

1002

IED SAN JOSEMARIA ESCRIVA DE BALAGUER

Tecnología e Informática

Chía, Cundinamarca

2014

INTRODUCCIÓN

Este trabajo tiene el objetivo de entrar en un aprendizaje profundo relacionado con algoritmos y fundamentos de programación, este tema se considera necesario para la creación de un juego educativo virtual, ya que este tipo de proceso permite establecer un orden que sea adecuado para un seguimiento, apoyándonos en diagramas de flujo para representar gráficamente un algoritmo o apoyándonos de un seudocódigo que emplea algún lenguaje usando algunas convenciones sintácticas.

La metodología que aplicamos para elaborar este taller es por medio de definiciones de conceptos básicos, identificación de características, técnicas que permitan la formulación, representaciones graficas, reglas estipuladas, relaciones entre conceptos. La metodología ha sido ofrecida por el docente, el cual estará también ayudándonos constantemente durante este proceso, pues al ser un tema nuevo se pueden generar algunas confusiones.

Por último los estudiantes tenemos que tener en cuenta que este trabajo es una de las bases que serán parte del apoyo para la realización del proyecto que tiene como objetivo la creación de un juego educativo, el lenguaje de programación, algoritmos, fundamentos de programación y otros temas que serán vistos próximamente son el principio constitutivo teórico para el desarrollo proactivo de este trabajo.

RESUMEN

En resumen, un algoritmo es cualquier cosa que funcione paso a paso, donde cada paso se pueda describir sin ambigüedad y sin hacer referencia a una computadora en particular, y además tiene un límite fijo en cuanto a la cantidad de datos que se pueden leer/escribir en un solo paso. Esta amplia definición abarca tanto a algoritmos prácticos como aquellos que solo funcionan en teoría, por ejemplo el método de Newton y la eliminación de Gauss-Jordan funcionan, al menos en principio, con números de precisión infinita; sin embargo no es posible programar la precisión infinita en una computadora, y no por ello dejan de ser algoritmos.

En particular es posible considerar una cuarta propiedad que puede ser usada para validar la tesis de Church-Turing de que toda función calculable se puede programar en una máquina de Turing (o equivalentemente, en un lenguaje de programación suficientemente general): Aritmetizabilidad: Solamente operaciones innegablemente calculables están disponibles en el paso inicial.

Recordemos que un algoritmo es utilizado en la vida cotidiana, ya que se emplean algoritmos frecuentemente para resolver problemas. Algunos ejemplos son los manuales de usuario, que muestran algoritmos para usar un aparato, o las instrucciones que recibe un trabajador por parte de su patrón.

OBJETIVOS

- Aprender y definir términos que son de gran importancia como fundamentos de programación, necesarios para la elaboración de un juego virtual educativo.
- Identificar las características y as técnicas de estos fundamentos de programación para poder darles un mejor uso, y prepararse para trabajar con ellos en la realización del proyecto creado por la institución.
- Identificar la simbología de los diagramas de flujo, que abren la posibilidad de entender con más facilidad que y como se maneja un algoritmo, en este diagrama de flujo se podrá ver el proceso adecuado que debe seguir el proyecto y en un futuro dará las órdenes respectivas al juego.
- Concientizarnos de cuales reglas se deben seguir para la elaboración de un diagrama de flujo ya que este será la estructura de rigiera el juego educativo, pues es el encargado de dictar las reglas.
- Relacionar conceptos con otros que aunque tienen definiciones diferentes se relacionan porque se encuentran englobados en el mismo tema de programación computacional.

PALABRAS CLAVE

Algoritmo: Es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad. Dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución. Los algoritmos son el objeto de estudio de la algoritmia.

Ciencias de la Computación: Las ciencias de la computación o ciencias computacionales son aquellas que abarcan las bases teóricas de la información y la computación, así como su aplicación en sistemas computacionales. Existen diversos campos o disciplinas dentro de las ciencias de la computación o ciencias computacionales; algunos resaltan los resultados específicos del cómputo (como los gráficos por computadora), mientras que otros (como la teoría de la complejidad computacional) se relacionan con propiedades de los algoritmos usados al realizar cómputos y otros se enfocan en los problemas que requieren la implementación de cómputos.

Lenguajes de Programación: Un lenguaje de programación es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana.

Diagrama de Flujo: El diagrama de flujo o diagrama de actividades es la representación gráfica del algoritmo o proceso. Se utiliza en disciplinas como programación, economía, procesos industriales y psicología cognitiva.

Seudocódigo: En ciencias de la computación, y análisis numérico, el pseudocódigo (o falso lenguaje) es una descripción de alto nivel compacta e informal del principio operativo de un programa informático u otro algoritmo.

DESARROLLO DEL TALLER DE ALGORITMOS Y FUNDAMENTOS DE PROGRAMACIÓN

1. Definir los siguientes términos:
    - **Algoritmo:** En matemáticas, lógica, ciencias de la computación y disciplinas relacionadas, un algoritmo es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad. Dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución. Los algoritmos son el objeto de estudio de la algoritmia.

      En la vida cotidiana, se emplean algoritmos frecuentemente para resolver problemas. Algunos ejemplos son los manuales de usuario, que muestran algoritmos para usar un aparato, o las instrucciones que recibe un trabajador por parte de su patrón. Algunos ejemplos en matemática son el algoritmo de multiplicación, para calcular el producto, el algoritmo de la división para calcular el cociente de dos números, el algoritmo de Euclides para obtener el máximo común divisor de dos enteros positivos, o el método de Gauss para resolver un sistema lineal de ecuaciones.
    - **Algoritmo Cualitativo:** Son aquellos en los que se describen los pasos utilizando palabras. Se describen los pasos utilizando palabras. Son aquellos en los que se describen o se resuelven problemas de la vida cotidiana pero siempre enmarcadas en tres estructuras fundamentales que son : Secuencias de acciones, Decisión de acción y Los Ciclos de acciones
    - **Algoritmo Cuantitativo:** Se utilizan cálculos numéricos para definir los pasos del proceso. Son aquellos en los que se utilizan cálculos numéricos para definir los pasos del proceso. De igual forma estos tipos de algoritmos describen tres partes esenciales: Entrada, Proceso y Salida. Es una serie de pasos organizados que describe el proceso que se debe seguir, para dar solución a un problema

especifico, son aquellos en los que se utilizan cálculos numéricos para definir los pasos del proceso.

- **Lenguaje Algorítmico:** Es una serie de símbolos y reglas que se utilizan para describir de manera explícita un proceso, que servirán de apoyo para describir las soluciones que aquí se plantean.
- Los algoritmos pueden describirse utilizando diversos lenguajes. Cada uno de estos lenguajes permiten describir los pasos con mayor o menor detalle. La clasificación de los lenguajes para algoritmos puede enunciarse de la siguiente manera :
  - Lenguaje Natural.
  - Lenguaje de Diagrama de Flujo.
  - Lenguaje Natural de Programación.
  - Lenguaje de Programación de Algoritmos.

  Teniendo en cuenta la forma en que describen el proceso, existen dos tipos de lenguajes algorítmicos principales y esto son los siguientes:
- **Lenguajes Grafico:** Es la representación gráfica de las operaciones que realiza un algoritmo (diagrama de flujo). Es decir que por medio de una grafica permite mostrar el algoritmo para que se manifieste visualmente la relación matemática o correlación estadística que guardan entre sí. También es el nombre de un conjunto de puntos que se plasman en coordenadas cartesianas y sirven para analizar el comportamiento de un proceso o un conjunto de elementos o signos que permiten la interpretación de un fenómeno. La representación gráfica permite establecer valores que no se han obtenido experimentalmente sino mediante la interpolación (lectura entre puntos) y la extrapolación (valores fuera del intervalo experimental).
- **Lenguaje No Grafico:** Representa en forma descriptiva las operaciones que debe realizar un algoritmo (pseudocódigo). En el pseudocódigo se mezcla el lenguaje de programación y un idioma como el español, que se emplea dentro de la programación estructurada, para especificar el diseño de un programa. Se puede definir como un lenguaje de especificaciones de algoritmos, utilizando palabras que indican el proceso a realizar.

2. ¿Cuáles son las características de un algoritmo?

La palabra algoritmo se deriva de la traducción al latín de la palabra árabe Alkhowarizmi, nombre de un matemático y astrónomo árabe que escribió un tratado sobre la manipulación de números y ecuaciones en el siglo IX. Se define como una serie de pasos organizados que describen el proceso que se debe seguir, para dar solución a un problema específico.

Las principales características que debe tener un buen algoritmo son:

- Debe tener un punto particular de inicio.

- Debe ser completamente definido y no debe permitir dobles interpretaciones.
- Debe ser general, es decir, soportar la mayoría de las variantes que se puedan presentar en la definición del problema.
- Debe ser finito en tamaño y tiempo de ejecución.
- Debe ser legible, claro y fácil de interpretar y entender.
- Debe ser realizable, el proceso algorítmico debe terminar después de una cantidad finita de pasos. Se dice que un algoritmo es inaplicable cuando se ejecuta con un conjunto de datos iniciales y el proceso resulta infinito o durante la ejecución se encuentra con un obstáculo insuperable sin arrojar un resultado.
- Debe ser comprensible, debe ser claro lo que hace, de forma que quien ejecute los pasos (ser humano o máquina) sepa qué, cómo y cuándo hacerlo. Debe existir un procedimiento que determine el proceso de ejecución.
- Debe ser preciso, el orden de ejecución de las instrucciones debe estar perfectamente indicado. Cuando se ejecuta varias veces, con los mismos datos iniciales, el resultado debe ser el mismo siempre. La precisión implica determinismo.

3. ¿Qué es Codificación?

   La codificación es la operación de escribir la solución del problema

(De acuerdo a la lógica del diagrama de flujo o pseudocódigo), en una serie de instrucciones detalladas, en un código reconocible por la computadora. Una vez que los algoritmos de una aplicación han sido diseñados, ya se puede iniciar la fase de codificación. En esta etapa se tienen que traducir dichos algoritmos a un lenguaje de programación específico; es decir, las acciones definidas en los algoritmos hay que convertirlas a instrucciones. Para codificar un algoritmo hay que conocer la sintaxis del lenguaje al que se va a traducir. Sin embargo, independientemente del lenguaje de programación en que esté escrito un programa, será su algoritmo el que determine su lógica. La lógica de un programa establece cuáles son sus acciones y en qué orden se deben ejecutar. Por tanto, es conveniente que todo programador aprenda a diseñar algoritmos antes de pasar a la fase de codificación. La serie de instrucciones escritas para un programa se les conoce como código fuente y se escriben en un lenguaje de programación que puede ser de bajo, medio o alto nivel. Diseño de algoritmos y su codificación en lenguaje C introduce los conceptos fundamentales de la programación, especialmente en el diseño de algoritmos, la programación estructurada y la codificación en lenguaje C. Con esta obra el lector recibe una orientación básica y fundamental en dicha área. Para traducir un algoritmo representado en UN pseudo-código, al lenguaje de programación C++, se deben seguir las siguientes reglas, que se ven en esta grafica:

	SEUDOCODIGO	C++
Definición de Variables	x:tipo	tipo x;
Asignación	:=	=
Operadores Aritméticos		
Suma	+	+
Resta	-	-
Multiplicación	*	*
División	/	/
Módulo	**mod**	%
Lectura	**leer**(a)	**cin** >>a;
Impresión	**escribir**(a)	**cout** << a;
Cambio de Línea	cambio_linea	'\n'
Cadena Carácteres	"cadena"	"cadena"

Selección	**si** condición
	entonces
	bloque_instrucciones1
	sino
	bloque_instrucciones2
	fin_si
		if (condición) {
		bloque_instrucciones1;
		} **else** {
		bloque_instrucciones2;
		};

Selección Múltiple	**Seleccionar** (opción) **de**
	caso constante 1:
	bloque_instrucciones_1;
	-
	-
	-
	caso constante n:
	bloque_instrucciones_n
	fin_seleccionar
		switch (opción){
		case constante_1:
		bloque_instrucciones1;
		break;
		.
		.
		case constante_n:
		bloque_instrucciones_n;
		break;
		};

Comentarios	/* comentario */	/* comentario */
Operadores Lógicos		
Negación	~	!
y lógico	&	&&
o Lógico	¡	¡¡
Operadores Relacionales		
Menor que	<	<
Mayor que	>	>
Igual a	=	==
Menor o igual que	<=	<=
Mayor o igual que	>=	>=
Diferente a	<>	!=

4. ¿Cuales son las técnicas para la formulación de algoritmos?

   Técnicas de diseño

   Top down:

   También conocida como de arriba-abajo y consiste en establecer una serie de niveles de mayor a menor complejidad (arriba-abajo) que den solución al problema. (Hernández, 2010)

   Bottom Up:

   El diseño ascendente se refiere a la identificación de aquellos procesos que necesitan procesarse en el momento en el que vayan apareciendo para satisfacer el problema inmediato. (Hernández, 2010)

5. ¿Qué son los Diagramas de flujo y los Pseudocodigos?

   - **Diagrama de Flujo**

     Es la representación detallada en forma gráfica de cómo deben realizarse los pasos en la computadora para obtener resultados. El diagrama de flujo o diagrama de actividades es la representación gráfica del algoritmo o proceso. Se utiliza en disciplinas como programación, economía, procesos industriales y psicología cognitiva.

     En Lenguaje Unificado de Modelado (UML), un diagrama de actividades representa los flujos de trabajo paso a paso de negocio y operacionales de los componentes en un sistema. Un diagrama de actividades muestra el flujo de control general.

     En SysML el diagrama de actividades ha sido extendido para indicar flujos entre pasos que mueven elementos físicos (p.ej., gasolina) o energía (p.ej., presión). Los cambios adicionales permiten al diagrama soportar mejor flujos de comportamiento y datos continuos.

     Estos diagramas utilizan símbolos con significados definidos que representan los pasos del algoritmo, y representan el flujo de ejecución mediante flechas que conectan los puntos de inicio y de fin de proceso.

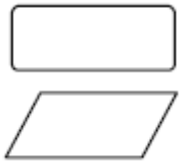
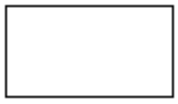
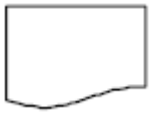
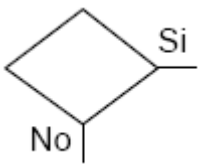
- **Pseudocódigo**

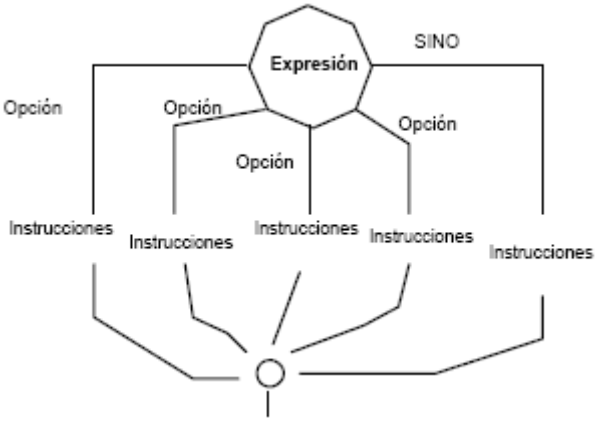
  Mezcla de lenguaje de programación y español (o inglés o cualquier otro idioma) que se emplea, dentro de la programación estructurada, para realizar el diseño de un programa. Si bien es cierto, podemos Sentarnos frente a la computadora y programar un algoritmo, pero lo anterior corresponde a un conjunto de buenas prácticas que debemos tomar antes de comenzar a programar. Dentro de estas buenas prácticas para el desarrollo de programas tenemos, por último, las pruebas de escritorio. Dichas pruebas nos permiten verificar de manera manual los valores que van obteniendo cada una de las variables involucradas en el programa, siguiendo la lógica de programación establecida. De esta forma, podemos tener mayor certidumbre de que el algoritmo al programarlo realizará lo que realmente queremos que haga. En ciencias de la computación, y análisis numérico, el pseudocódigo (o falso lenguaje) es una descripción de alto nivel compacta e informal del principio operativo de un programa informático u otro algoritmo.

  Utiliza las convenciones estructurales de un lenguaje de programación real, pero está diseñado para la lectura humana en lugar de la lectura mediante máquina, y con independencia de cualquier otro lenguaje de programación. Normalmente, el pseudocódigo omite detalles que no son esenciales para la comprensión humana del algoritmo, tales como declaraciones de variables, código específico del sistema y algunas subrutinas. El lenguaje de programación se complementa, donde sea conveniente, con descripciones detalladas en lenguaje natural, o con notación matemática compacta. Se utiliza pseudocódigo pues este es más fácil de entender para las personas que el código del lenguaje de programación convencional, ya que es una descripción eficiente y con un entorno independiente de los principios fundamentales de un algoritmo. Se utiliza comúnmente en los libros de texto y publicaciones científicas que se documentan varios algoritmos, y también en la planificación del desarrollo de programas informáticos, para esbozar la estructura del programa antes de realizar la efectiva

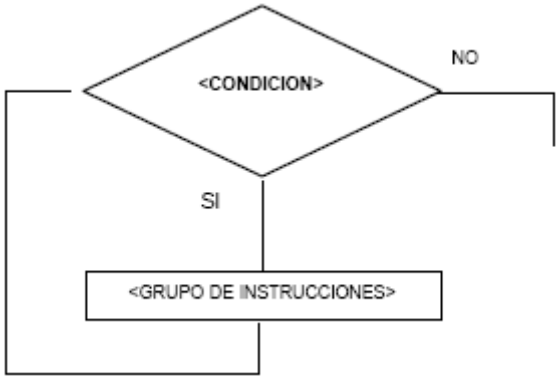
codificación. No existe una sintaxis estándar para el pseudocódigo, aunque los ocho IDE's que manejan pseudocódigo tengan su sintaxis propia. Aunque sea parecido, el pseudocódigo no debe confundirse con los programas esqueleto que incluyen código ficticio, que pueden ser compilados sin errores. Los diagramas de flujo y UML pueden ser considerados como una alternativa gráfica al pseudocódigo, aunque sean más amplios en papel.

6. Graficar la simbología de los diagramas de flujo y relacionar cada símbolo con su función.

Símbolo	Descripción
	Indica el inicio y el final de nuestro diagrama de flujo.
	Indica la entrada y salida de datos.
	Símbolo de proceso y nos indica la asignación de un valor en la memoria y/o la ejecución de una operación aritmética.
	Indica la salida de información por impresora.
	Conector dentro de página. Representa la continuidad del diagrama dentro de la misma página.
	Conector fuera de página. Representa la continuidad del diagrama en otra página.
	Indica la salida de información en la pantalla o monitor.
	Símbolo de decisión. Indica la realización de una comparación de valores.

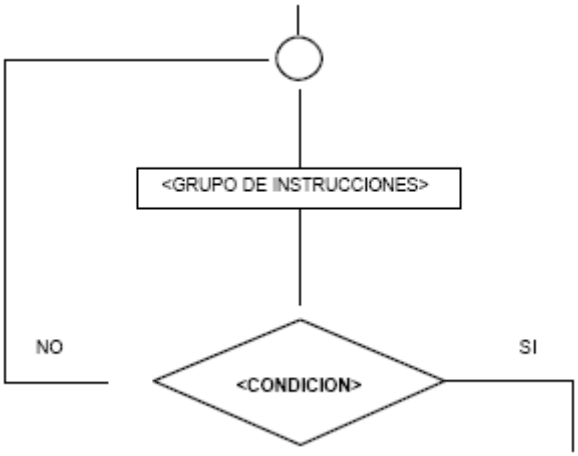


Símbolo de Selección Múltiple. Dada una expresión permite escoger una opción de muchas.

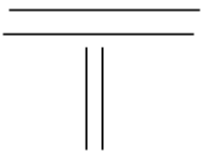


Símbolo del Mientras. Dada una expresión al principio de la iteración esta es evaluada; si la condición es verdadera realizará el ciclo, si es falsa la repetición cesará.

Símbolo del Para. Esta estructura de control repetitiva se usa generalmente cuando se conoce de antemano el número de iteraciones.



Símbolo Repita Hasta. Funciona igual que la estructura Mientras, con la diferencia que al menos una vez hará el grupo de instrucciones y luego evaluará una condición. Si la condición evaluada es falsa continua dentro del ciclo y si es verdadera termina la iteración.



Líneas de flujo o dirección. Indican la secuencia en que se realizan las operaciones.

7. Cuáles son las reglas para la elaboración de un diagrama de flujo.

• Poner un encabezado que incluya un título que identifique la función del algoritmo; el nombre del autor; y la fecha de elaboración.

• Sólo se pueden utilizar símbolos estándar (ISO 5807).

• Los diagramas se deben dibujar de arriba hacia abajo y de izquierda a derecha.

• La ejecución del programa siempre empieza en la parte superior del diagrama.

• Los símbolos de “Inicio” y “Final” deben aparecer solo una vez;

• La dirección del flujo se debe representar por medio de flechas (líneas de flujo).

• Todas las líneas de flujo deben llegar a un símbolo o a otra línea;

• Una línea de flujo recta nunca debe cruzar a otra. Cuando dos líneas de flujo se crucen, una de ellas debe incluir una línea arqueada en el sitio donde cruza a la otra (ilustración 2-5).

• Se deben inicializar las variables que se utilicen o permitir la asignación de valores mediante consulta al usuario.

• Las bifurcaciones y ciclos se deben dibujar procurando una cierta simetría.

• Cada rombo de decisión debe tener al menos dos líneas de salida (una para SI y otra para NO).

• Las acciones y decisiones se deben describir utilizando el menor número de palabras posible; sin que resulten confusas o poco claras.

• Si el Diagrama se vuelve complejo y confuso, es mejor utilizar símbolos conectores para reducir las líneas de flujo.

• Todo el Diagrama debe ser claro, ordenado y fácil de recorrer.

• El Diagrama se debe probar recorriéndolo con datos iniciales simples (prueba de escritorio).

8. Definir los siguientes términos:
    - Variable: En programación, una variable está formada por un espacio en el sistema de almacenaje (memoria principal de un ordenador) y un nombre simbólico (un identificador) que está asociado a dicho espacio. Ese espacio contiene una cantidad o información conocida o desconocida, es decir un valor. El nombre de la variable es la forma usual de referirse al valor almacenado: esta separación entre nombre y contenido permite que el nombre sea usado independientemente de la información exacta que represente.
    - Constante: En programación, una constante es un valor que no puede ser alterado durante la ejecución de un programa.

      Una constante corresponde a una longitud fija de un área reservada en la memoria principal del ordenador, donde el programa almacena valores fijos.

      Por ejemplo:

      El valor de pi= 3.1416

      Por conveniencia, el nombre de las constantes suele escribirse en mayúsculas en la mayoría de lenguajes.
    - Contador: Los contadores en MicroMundos se implementan como una estructura de programación (da “A :A + 1) que consistente en almacenar en una variable (“A) el valor de ella misma (:A) más un valor constante (1). Es muy útil para controlar el número de veces que debe ejecutarse un grupo de instrucciones.

      En Scratch, se utiliza la instrucción cambiar ... por ...para incrementar la variable en una cantidad determinada.
    - Acumulador: Estructura muy utilizada en programación (da “A :A + :B) y que consiste en almacenar en una variable (“A) el valor de ella misma (:A) más otro valor variable (:B). Es muy útil para calcular sumatorias.
    - Identificador: Los identificadores son nombres que se dan a los elementos utilizados para resolver un problema y poder diferenciar unos de otro

CONCLUSIONES

1. Los algoritmos son un conjunto de reglas ordenadas que nos permiten realizar una actividad por medio de pasos que permiten que no se creen dudas a quien deba estas actividades.
2. Los algoritmos son el objeto de estudio de la algoritmia.
3. Los algoritmos son necesarios para la creación de juegos virtual.
4. Los algoritmos los podemos utilizar en nuestra vida cotidiana pues estos nos permiten resolver diversos problemas, por ejemplo los manuales de usuario, que nos proporcionan las instrucciones necesarias para usar un aparato tecnológico

REFERENCIAS BIBLIOGRAFICAS

1. Algoritmos. 2014
   https://sites.google.com/a/unal.edu.co/algoritmos-2014-1/
   Sites.google.com-Colombia
2. Fundamentos de programación. 2002
   http://es.wikibooks.org/wiki/Fundamentos_de_programaci%C3%B3n
   Wikibooks.org-Colombia
3. Algoritmos. 2013
   http://es.wikipedia.org/wiki/Algoritmo
   Wikipedia.org-Colombia