METODOLOGÍA PARA LA SOLUCIÓN DE PROBLEMAS ALGORÍTMICOS

JOSE FRANCISCO AMADOR MONTAÑO*
JOHN ALEXANDER ROJAS MONTERO**

RESUMEN

La solución de problemas algorítmicos es un tema que concierne a muchas especialidades. En "ciencias de la computación" es necesario acordar una metodología para la solución de dichos problemas, con el propósito de mantener un lenguaje común a la hora de abordarlas.

* Licenciado en Matemáticas, Universidad Pedagógica Nacional
Especialista en Gerencia y Gestión Cultural, Colegio Mayor de Nuestra Señora del Rosario
Ingeniero de Sistemas, Escuela de Administración de Negocios.
Docente de tiempo completo EAN.

** Ingeniero de Sistemas, Universidad Nacional de Colombia
Maestría "Tecnologías de la Información Aplicadas a la Educación", Universidad Pedagógica Nacional
Docente de tiempo completo EAN.

INTRODUCCIÓN

El Departamento de Ciencias de la Computación de la E.A.N., esta conformando una metodología que favorezca la solución de problemas que tengan características algorítmicas, con el propósito de brindar apoyo a la Ingeniería de Software en el campo de la programación, para que se tengan procesos formales desde el momento del análisis hasta la implementación, buscando un permanente contacto con las Ciencias Básicas con el fin de hacer de este proceso una labor profesional y formal.

La programación es un área que se encuentra en una etapa crítica, porque sus procesos no tienen métodos formales que garanticen la efectividad de un producto. Aún más, no hay manera de predecir el real funcionamiento de un programa bajo condiciones exigentes y apremiantes. Esta situación crece de manera crítica cuando los sistemas distribuidos se popularizan. Con facilidad encontramos por ejemplo bancos que ofrecen transacciones que no se pueden realizar en determinados momentos, reservas de vuelos asignadas en otras horas y a otras personas, lento pago de impuestos, inconvenientes que causa la mala facturación de los servicios públicos, dificultades en el traslado de historias médicas de institución a institución y sus respectivas pérdidas de datos,etc.

Resulta fácil prever que la programación está siendo exigida para un mayor número de actividades de una complejidad superior, lo que lleva a que el software hecho hoy ya mañana resulte obsoleto. Peor aún, los programas que intentan virtualizar la realidad son rechazados por ella misma, dado el permanente cambio de condiciones.

Debido a esta situación los productos de software se han convertido en grandes paquetes inmanejables e incontrolables. A decir verdad, los proyectos en software muchas veces encuentran mejor camino en la iniciación desde cero que en la actualización o en su debida complementación.[1]

La industria apoya los trabajos que favorezcan la automatización y sistematización de procesos donde el hombre tiene altos márgenes de error pero resulta desconcertante que la solución de problemas conlleve a una mayor problemática. Pareciera que la espiral crece en complejidad y que los problemas no se solucionan del todo.

Al presentarse un problema de alta complejidad en un programa, surgen otros, que son, más un paliativo que una solución eficaz y procedente. El usuario o los usuarios encuentran en los productos de software grandes impedimentos que no hay forma de predecir, porque su solución no está en manuales y porque los autores del producto no tienen la real dimensión de la situación; por eso con frecuencia al usuario se le exige que se adapte al sistema cuando el sistema debe ser lo suficientemente elaborado para que el usuario y el producto tengan una relación natural.

Razón por la cual se hace necesaria la implementación de métodos cuantitativos y métodos formales para dimensionar el comportamiento de los programas en el mundo real; según Gibbs "Por desdicha las matemáticas tradicionales aptas para la descripción de sistemas físicos no son aplicables al universo sintético binario de un computador; es la matemática discreta una especialidad mucho menos madura la que gobierna este campo, aún así valiéndose del instrumental no muy amplio todavía de la teoría de conjuntos y del cálculo de predicados los informáticos se las han arreglado para traducir especificaciones y programas al lenguaje matemático, donde pueden analizarse con los instrumentos teóricos denominados métodos formales".

Con frecuencia los programas de computador tienen deficiencias en: **seguridad**: a cada nueva estrategia que persiga brindar mayor confianza a la reserva en el uso del software, surgen otras estrategias que la contrarrestan, proceso que lleva a la piratería y a la pérdida de la intimidad en el manejo de la información. **Calidad del software**: no cuenta con indicadores que posibiliten el justiprecio de estos productos. **Garantías:** no dan seguridad en la solución de problemas que surjan en el futuro ya que no se conocen. **Costo**: no es calculable desde un principio.

Esta problemática se esta enfrentando con *métodos evolutivos* que se adaptan a las contingen-

[1] Gibbs, Wait. "La crisis crónica de la programación", en Investigación y Ciencia, Noviembre(1994).

cias de la realidad y se hace necesario que la programación vincule a sus formas de trabajo *metodologías ingenieriles* que han sido puestas en práctica y han dado buenos resultados a través de la historia.

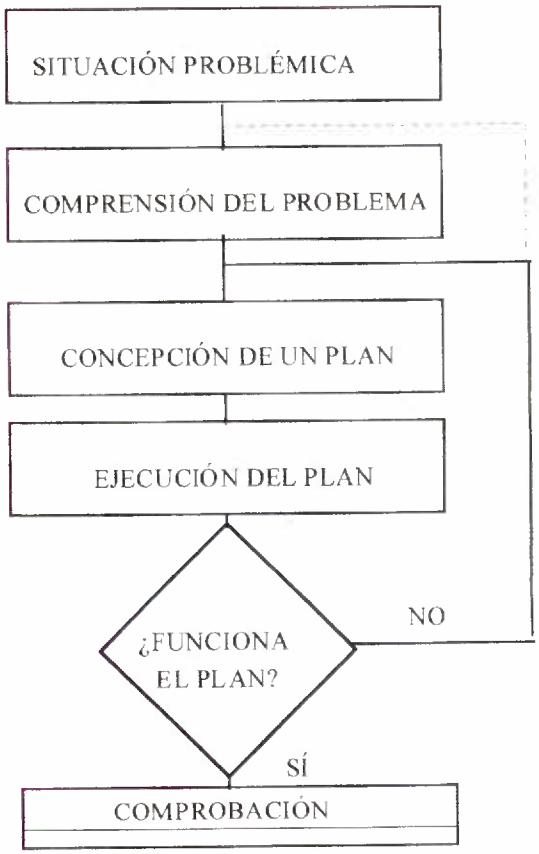
1. MARCO TEÓRICO

1.1 La solución de problemas

Como fundamento teórico nos hemos apoyado en los principios de la psicología cognitiva con énfasis en los procesos de pensamiento para resolver problemas matemáticos,que a su vez favorecen de manera notoria los procesos de pensamiento para resolver problemas algorítmicos. Por esta razón consideraremos en nuestro marco teórico las siguientes categorías de conocimiento según Mayer[2]:

- Conocimiento lingüístico
- Conocimiento semántico
- Conocimiento esquemático
- Conocimiento operativo
- Conocimiento estratégico

Desde este punto de vista el aprendizaje de las matemáticas se convierte en una piedra angular para modelar la realidad circundante con la cual se construirán nuevos conocimientos, que tendrán que ser utilizados para resolver nuevos problemas. Para ello se puede contar con esquemas de solución de problemas como el ofrecido por Polya[3]:



De esta manera encontramos que el conocimiento lingüístico, semántico y esquemático nos permiten determinar qué situación es problemática, la que debe ser comprendida en su totalidad por medio del conocimiento específico, la experiencia y los modelos o esquemas existentes.

Tanto el conocimiento operativo, como el estratégico favorecen la formulación de planes y metodologías para buscar la solución del problema. Este proceso debe ser verificado permanentemente para que las condiciones iniciales garanticen procesos coherentes y soluciones consecuentes.

1.2 La Disciplina de la Computación

Con frecuencia se asocia el ejercicio de las ciencias de la computación con programación de computadores, desconociendo la existencia de otras estructuras necesarias para la organización de sistemas complejos e interactivos que modelan la realidad. Es necesario que tanto la teoría como el ejercicio en el campo de las ciencias de la computación se integren y den como resultado procesos estructurados que puedan ser comprobados, verificados y optimizados por métodos científicos.

[2] Mayer, Richard E. (1986). "Pensamiento, Solución de Problemas y Cognición." 1a.ed. Barcelona: Ediciones Paidos. p. 408.

[3] Polya, G. (1965). "Cómo plantear y resolver problemas: un nuevo enfoque del método matemático.". México: Editorial Trillas.

Y ¿qué se entiende por la profesión de las ciencias de la computación?. En sentido amplio [4] "profesión es un conjunto formado por la gente, las tecnologías, las instituciones y las prácticas que cuidan de los intereses de las personas y de los hitos del dominio". Entendiéndose por dominio los intereses continuos y por hito las rupturas o crisis que alteran el desarrollo normal de las acciones.

Atendiendo a la definición de profesionales en ciencias de la computación es necesario reconocer los paradigmas que favorecen su competencia frente a los hitos e intereses humanos, los cuales son bastiones del pensamiento que conforman la disciplina; en este fenómeno encontramos en Tucker una aproximación a ellos:

"El paradigma de la teoría tiene sus raíces en las matemáticas y la lógica, cuyo legado nos permite trabajar con algoritmos complejos y sutiles. El paradigma de la abstracción tiene sus raíces en la larga tradición del método científico, cuyo legado nos permite formular y probar hipótesis sobre algoritmos, máquinas y modelos. El paradigma del diseño tiene sus raíces en la larga tradición de la ingeniería, cuyo legado nos capacita para diseñar máquinas capaces de calcular correctamente, y de procesar información en dominios en los que se desarrolla el trabajo humano".[5]

En Algoritmos y Estructuras de datos estos paradigmas se presentan en el Cuadro N° 1:[6]

1.3 Solución de problemas algorítmicos mediante la utilización del computador

Los problemas que se resuelven mediante la utilización del computador se encuentran en tres áreas especialmente, a saber:

· **Matemática**: donde se requiere de un manejo adecuado de sistemas numéricos, sistemas lógicos y en general de la fundamentación matemática.

· **Gráfica**: donde son necesarias nociones y habilidades geométricas para la buena manipulación de objetos de esta naturaleza.

· **Textual**: donde se hace imprescindible tener dominio sobre las unidades de información básicas, como son los caracteres del lenguaje escrito y las palabras.

El abordar los problemas teniendo en cuenta estas tres áreas, ofrece un espacio de relaciones, métodos y principios que se integran a la hora de

Cuadro N° 1

Teoría	Abstracción	Diseño
Computabilidad Complejidad computacional Límites espacio - tiempo para algoritmos Niveles de intratabilidad Computación paralela Representación de los algoritmos Caminos para comunicación en la máquina Algoritmos probabilísticos Teoría de grafos Funciones recursivas Relaciones de recurrencia Combinatorias Cálculo Inducción Lógica de predicados y temporal Semánticas Probabilidad Estadística	Eficiencia, optimización y verificación de algoritmos Análisis de la solución mejor, peor y promedio Clasificaciones de los efectos de control y las estructuras de datos en los requerimientos de tiempo y espacio Importantes clases de técnicas Algoritmos distribuidos y paralelos Métodos de partición de problemas en tareas ejecutables en procesadores separados	Selección, implementación y prueba de algoritmos Implementación y prueba de métodos generales Algoritmos distribuidos Manejadores de almacenamiento Prueba experimental de algoritmos heurísticos para problemas combinacionales Protocolos criptográficos

[4] Denning, Peter J. (1994). En el prólogo de "Fundamentos de Informática: Lógica, solución de problemas, programas y computadoras." de Tucker, Allen B. y otros. México: McGraw Hill, 1994. 392 p.

[5] Tucker, Allen B.; Cupper, Robert D.; Bradley,W. James; Garnick, David K. Fundamentos de Informática: Lógica, solución de problemas, programas y computadoras. México: McGraw Hill, 1994. p, xviii.

[6] Denning, P.; Comer, D.; Mulder, M.; Tucker, A.; Turner, A., y Young, P. "Computing as a Discipline", Report of the ACM Task Force on the Core of Computer Science. En: Computer, February, 1989.

resolver problemas. A esta integración se le conoce como **solución de problemas algorítmicos**.

Para comenzar a tratar el tema de la solución de problemas algorítmicos es necesario tener en cuenta las siguientes definiciones:

Se entiende por algoritmo una lista de instrucciones que realizan una descripción paso a paso y precisa de un proceso que garantiza que resuelve cualquier problema que pertenezca a un tipo determinado, y que termina después de que se hayan llevado a cabo un número finito de pasos[7].

Es importante recordar las características que Donald Knuth establece para un algoritmo: finitud, no ambigüedad, entradas/salidas y efectividad.

A partir de este momento se entenderá por problema algorítmico, cualquier problema conceptual o práctico cuya solución puede expresarse mediante un algoritmo[8].

Ahora bien, cuando se lleva un algoritmo a un computador con el propósito de observar la solución de un problema, el cual subyace a dicho algoritmo, es conveniente tener en cuenta que todo lo que haga un equipo de cómputo es realizable por el hombre, claro está, si dispone de tiempo y de los recursos necesarios para hacer una ingente cantidad de cálculos y operaciones.

El modelo computacional que se tiene para resolver problemas mediante el uso de computador es:

En dicho modelo, la entrada representa los datos que se deben suministrar o que se tenían antes de resolver el problema; dichos datos son necesarios para el funcionamiento o ejecución de los pasos del algoritmo, y se accesan por medio de dispositivos de entrada (teclado, ratón, micrófono, escáner, etc.).

El proceso son los pasos previstos en el algoritmo para llegar a la solución, en el computador se asocia esto con el procesador y la memoria donde se realizan dichos pasos.

La salida es el resultado solución del problema que puede ser observado por cualquiera de los dispositivos de salida con que se cuente (monitor, parlantes, impresora, etc.).

Dicho modelo lleva a considerar un modelo similar en el computador:

Para aplicar este modelo de entrada - proceso - salida a problemas algorítmicos es recomendable considerar las siguientes suposiciones:

i) Qué el número de entradas sea determinado y finito.

ii) Qué las entradas deben llegar al computador de forma física para ser procesadas.

iii) Que los pasos del proceso sean realizados en una secuencia adecuada y ordenada según el algoritmo.

iv) Que los pasos sean posibles de realizar en el computador, es este caso operaciones e instrucciones.

1.4 Descripción de algoritmos

Para resolver problemas algorítmicos es necesario establecer un lenguaje de descripción que permita encontrar una manera sintética y formal para describirlos. Este lenguaje tendrá en cuenta los estados inicial, final e intermedio, los que dará énfasis al modelo computacional:

---

[7] Tucker, Allen B.; Cupper, Robert D.; Bradley,W. James; Garnick, David K. "Fundamentos de Informática: Lógica, solución de problemas, programas y computadoras". México: McGraw Hill, 1994. p. 106.

[8] Tucker, Allen B.; Cupper, Robert D.; Bradley,W. James; Garnick, David K. "Fundamentos de Informática: Lógica, solución de problemas, programas y computadoras". México: McGraw Hill, 1994. p. 106.

1.4.1 Estados inicial y final de un algoritmo

Estos estados muestran intuitivamente la entrada y la salida en un algoritmo, los cuales son necesarios para precisar las condiciones que el algoritmo requiere tanto para su ejecución como para su solución.

Es posible asociar la descripción de las entradas al algoritmo con su estado inicial, es decir, antes de su ejecución, a esta descripción se le llamará **precondición**; igual ocurre con las salidas o el estado final, que se observa después de la ejecución del algoritmo, a la descripción de este estado final se le llamará **postcondición**.

Para describir tanto la precondición como la postcondición se acudirá a la lógica matemática con el propósito de garantizar el formalismo necesario que lleva al cumplimiento de las características de un algoritmo.

Al utilizar la lógica matemática para hacer estas descripciones de algoritmos concisas y precisas se acudirá a la lógica de predicados, y se llamarán **asertos**.

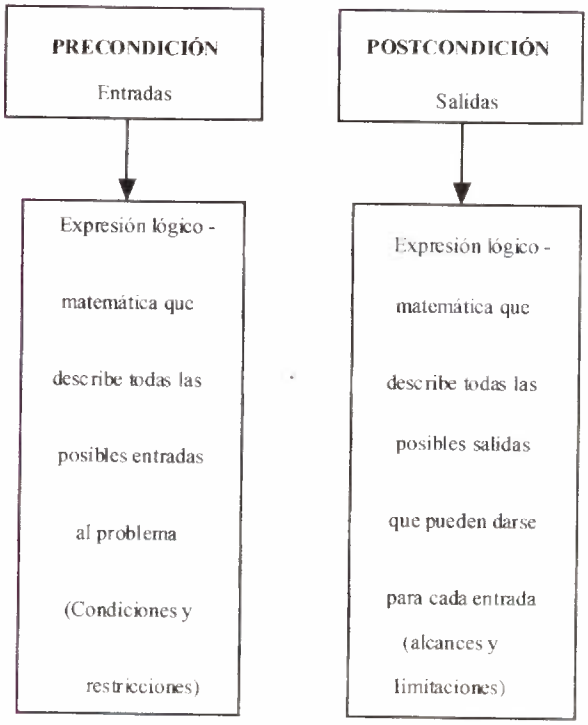
1.4.2 Estados intermedios de un algoritmo

Los estados intermedios de un algoritmo permiten:

i) Comprender el efecto de cada uno de los pasos sobre las variables que hacen parte del proceso que describe el algoritmo.

ii) Son un medio para verificar la veracidad y consistencia entre la precondición y la postcondición.

De esta manera, podemos considerar que la prueba de escritorio de un algoritmo que llegue hasta una comprobación generalizada, es la visualización de los estados intermedios de un algoritmo. Cada valor que se dé a las variables serán considerados como asertos de estos estados intermedios, pues son lógicamente aceptados.



2. Metodología para la solución de problemas algorítmicos

El siguiente esquema representa la metodología para la solución de problemas algorítmicos:

Esta metodología se apoya en el trabajo de MAPS[9] (Methodology for Algorithmic Problem Solving), el cual está adaptado para los estudiantes de ingeniería en procura de familiarizarlos con estos procesos, además de comenzarlos en la disciplina de la ingeniería de software.

2.1 Análisis del problema

El análisis es una etapa en la que se entrelazan los conocimientos lingüístico, semántico, operativo, esquemático y estratégico, con el propósito de tener claridad del problema y de mantener adecuadamente la información que se utilizará en cada estadio de esta metodología. Así, que para abordar el análisis de un problema resulta

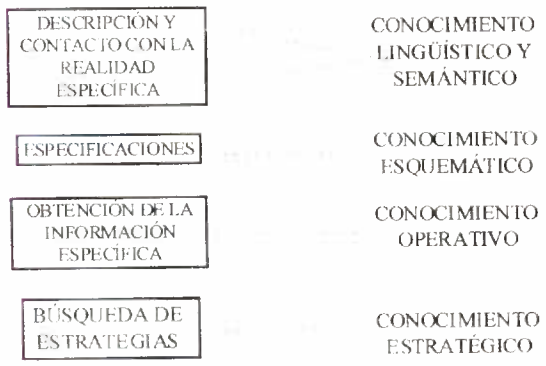
[9] TUCKER, Allen B.; Cupper, Robert D.; Bradley,W. James; Garnick, David K. Fundamentos de Informática: Lógica, solución de problemas, programas y computadoras. México: McGraw Hill, 1994. p. 147-149

indispensable contar con buena información acerca del mismo, además de práctica en la organización del conocimiento para que se puedan aterrizar las ideas que se tienen de la situación problemática y que ella pueda ser tratada con objetividad y solucionada parcial o totalmente en un período de tiempo determinado.

El análisis de un problema se abordará mediante procesos que involucran ciertos tipos de conocimientos, en la siguiente forma:

2.1.1 Descripción del problema

Para describir un problema es necesario determinar cuándo una situación es un problema para después formularlo con el propósito de tener claridad sobre el mismo, es decir, reconocer sus alcances y limitaciones internas y externas.



2.1.1.1 Determinación del problema

Un problema existe gracias a los propósitos, motivaciones y aspiraciones de los seres humanos. Es decir, una situación es problemática solamente cuando el ser humano la determina como tal[10].

Con frecuencia se utiliza el método científico para resolver problemas, esto es, aplicar el pensamiento deductivo e inductivo de forma combinada, siguiendo una serie de etapas sistemáticamente, con el propósito de dar respuesta a los interrogantes que la situación problemática presenta y no a otros.

Inicialmente se identificará una situación problema por medio de la percepción sensorial, así, es imprescindible que todos los sentidos tengan parte en este reconocimiento para tener una idea más real de lo que se quiere solucionar, se requiere experimentar el ambiente que rodea el problema, si es necesario se tendrá que acudir a las acciones de los sentidos como oler, tocar, ver, degustar y escuchar.

Con esta información sensorial el pensamiento deductivo e inductivo lleva a describir un problema en los términos que fue apropiado por el ser humano. Por ésta razón un problema tiene tantas interpretaciones como seres humanos lo hayan concebido así, con el propósito de unificar la concepción de un problema es necesario realizar una formulación de éste.

2.1.1.2 Formulación del problema

Al formular un problema se deben tener en cuenta los siguientes aspectos:

- Estar escrito en términos comprensibles para quien va a resolverlo, es decir, que no causen ambigüedades.
- Descartar el material intuitivo.
- Apoyar con gráficos que posibiliten captar las dimensiones del problema.

A partir de este momento es necesario definir con claridad el fin que se busca con la solución para no resolver otra situación problemática distinta a la formulada.

2.1.2 Especificaciones

Las especificaciones son el reflejo del conocimiento preciso del problema, es decir las entradas y las salidas son exactamente las necesarias y suficientes, son un esquema que representa concretamente los estados inicial, intermedio y final. Como se ve ofrecen un panorama global del problema, sus condiciones y su solución.

A partir de este instante se deben buscar esquemas de solución de problemas que tengan relación con el problema a resolver, a través de otros ya resueltos o de modelos que favorezcan la solución.

[10] Nadler, G. Hibino, S. and Farrel, J. Creative Solution Finding. Rockiln, CA: Prima Publishing. 1995.

Los esquemas de solución serán posteriormente representados en la etapa de diseño, por medio de diagramas de flujo, seudocódigo y cartas N - S, los cuales permiten reflejar los esquemas adoptados y adaptados al problema en particular. Es un lenguaje que debe ser ejercitado permanentemente con el fin de lograr representaciones precisas.

2.1.3 Obtención de la información específica

Una vez descrito el problema se inicia la tarea de buscar la información específica que permita orientar la ejecución de los procesos que se formaron por medio de las diferentes formas de representación, las que se confrontarán con el fin o los objetivos de la solución.

La búsqueda de información específica es exitosa en la medida en que las percepciones del problema sean objetivas, esto debido a que de esta manera se facilita la consecución de información en las fuentes más idóneas.

La confrontación de la información específica debe reflejar una permanente consecuencia entre ésta y los términos que componen la formulación del problema.

2.1.4 Búsqueda de estrategias

Con frecuencia los problemas algorítmicos se asocian a problemas matemáticos, por lo tanto muchas de las estrategias empleadas en dichas situaciones podrán ser utilizadas para enfrentar los primeros.

Las estrategias exigen de conocimiento específico, en este caso de algoritmia, para reconocer en el paso anterior la validación del siguiente, así desde el primero hasta la solución del problema.

2.2 Diseño del algoritmo

El diseño de la solución de problemas algorítmicos es el bosquejo que orienta cada uno de los procesos a considerar en la puesta en marcha de las estrategias acogidas.

2.2.1 Modularización

Es una estrategia que se utiliza con frecuencia en la solución de problemas algorítmicos. Ésta consiste en dividir el problema en módulos, los cuales son unidades independientes e interactuantes que al ser integrados proporcionan un camino fiable para llegar del estado inicial al final de las especificaciones del problema.

2.2.2 Definición de abstracciones

Las abstracciones son el conocimiento que permite o posibilita la comunicación entre módulos (variables, funciones, operaciones y esquemas).

En el diseño de la solución de problemas algorítmicos cuenta en gran medida la creatividad ya que esta ofrece un espacio constructivo donde las abstracciones y los módulos puedan mostrar diferentes arquitecturas, las cuales estarán vinculadas a lo clásico, lo moderno, lo alternativo y prospectivo. Esta etapa en la solución de problemas algorítmicos favorece la dimensión estética del ser humano.

2.2.3 Representaciones

Como se dijo en la sección "Especificaciones", para representar la solución de problemas algorítmicos se puede contar con:

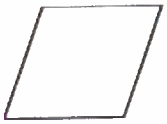
2.2.3.1 Diagramas de flujo

Símbolos unidos por flechas que representan el flujo de los procesos a seguir en la solución de problemas.

A continuación se puede observar los símbolos utilizados en los diagramas de flujo[11]:

Inicio/Fin:

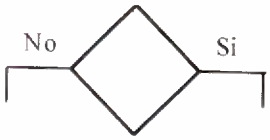
Entrada:



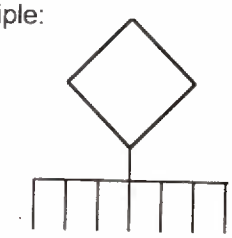
[11] Joyanes Aguilar, Luis. Fundamentos de programación: Algoritmos y Estructura de Datos. Colombia: McGraw-Hill, 1997. p. 47.

Proceso:

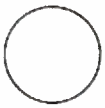
Decisión:



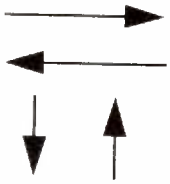
Decisión múltiple:



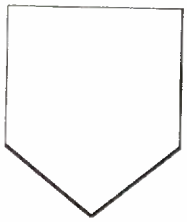
Conector misma página:



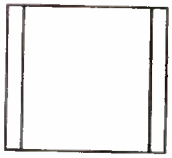
Indicador de dirección:



Conector página diferente

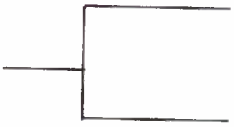


Módulo:



Salida:

Comentario:



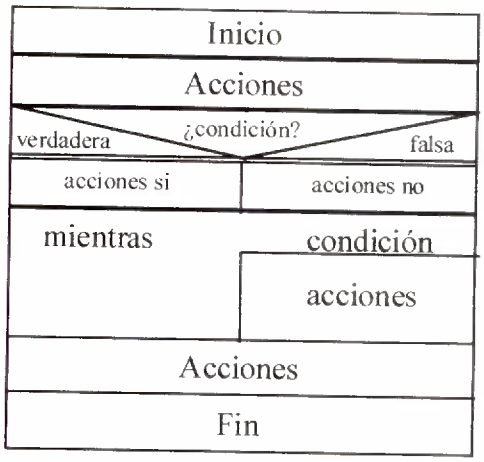
2.2.3.2 Cartas N - S

Las cartas N - S son una representación de los diagramas de flujo pero enmarcados en una caja, donde se muestra por niveles los procesos.

A continuación una carta N - S y sus partes[12]:

2.2.3.3 Seudocódigo

El seudocódigo posibilita la traducción de un lenguaje a otro (en este caso a un lenguaje de programación)[13].



[12] Joyanes Aguilar, Luis. Fundamentos de programación: Algoritmos y Estructura de Datos. Colombia: McGraw-Hill, 1997. p. 100-128.

[13] Bernal R., L. Alejandro. Lenguaje de Pseudo Código en Español de la E.A.N.: LES. (En impresión).

A continuación se observan las instrucciones básicas que se utilizarán en el seudocódigo para representar la solución de problemas algorítmicos:

```
Inicio

  Acciones;

  si (condición) entonces
        acciones_si;
  si_no
        acciones_no;

  mientras_que (condición) haga
        acciones;
  fin_mientras_que

  Acciones;

Fin
```

2.2.4 Prueba de escritorio

Para garantizar que la solución diseñada es correcta y completa en todos sus extremos, se debe probar y verificar, es decir, que al ingresar los datos de entrada permitidos por las precondiciones al algoritmo, se producirá un resultado consistente con la precondiciones.

2.3 Implementación de la solución

Después de haber realizado el Análisis y Diseño de la solución del problema se llega a la etapa de implementación, la cual se lleva a cabo en las siguientes fases:

2.3.1 Codificación

Dependiendo de la especialidad de quien resuelve el problema, se contarán con herramientas específicas para ello. En el caso de "ciencias de la computación" se cuenta con los lenguajes de programación, los cuales tienen su propia sintaxis. Por lo tanto se tendrán que hacer ajustes para transferir la representación de la solución del problema algorítmico a un lenguaje de éstos, además se podrá llegar a acuerdos en su presentación.

2.3.2 Prueba y verificación

Mediante la ejecución del programa en un computador es posible probar la efectividad del diseño ya implementado en un sistema de cómputo automático. Es necesario reconocer los errores típicos en este proceso porque de esta manera se podrá tener mejor contacto y comunicación con el lenguaje de programación.

2.4 Conclusiones

- Es necesario informar sobre las innovaciones realizadas para lograr el algoritmo solución (creatividad para resolver el problema), las herramientas necesarias (otros algoritmos, funciones, librerías, fragmentos de textos, apoyo de compañeros o profesionales).
- Nuevas experiencias, dificultades para la obtención del problema y su solución.
- Sustento matemático de la solución.
- Tiempo aproximado de ejecución y compilación (Bajo qué equipo).

2.5 Recomendaciones

- Son comentarios que apuntan a sugerir a personas que se enfrentan al mismo problema o a problemas similares.
- En qué otros problemas se puede utilizar esta solución y su algoritmo.
- Límites y alcances del trabajo.

3. Ejemplo

El siguiente ejemplo muestra varias etapas de la metodología de solución de problemas algorítmicos.

3.1 Análisis del problema

Formulación del problema

Se busca encontrar la suma de una cantidad determinada de números reales, cada vez que se quiera hacer esta operación.

Especificaciones

{ pre: entrada = ingresar la cantidad de números a sumar (n pertenece a los naturales y n diferente de cero), e ingresar los números (x pertenece a los reales) }

{pos: salida = entrega la suma de los números ingresados (y pertenece a los reales) }

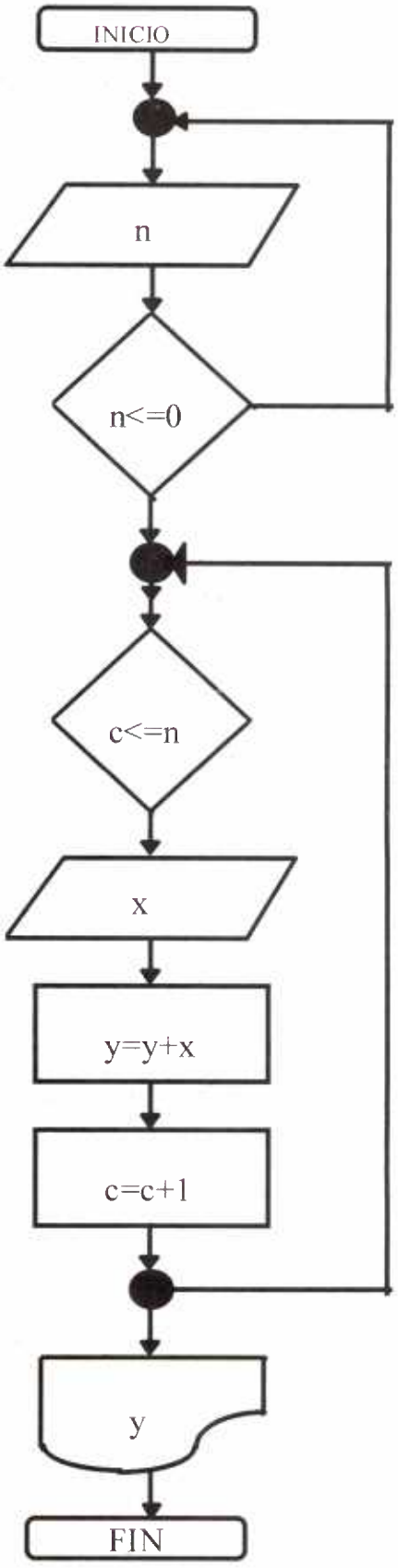
3.2 DISEÑO DEL ALGORITMO

Modularización

Paso 1: Ingresar la cantidad de números a sumar (verificando la precondición n pertenece a los naturales y n diferente de cero)
Paso 2: Calcular la suma de los números
Paso 2.1: Verificar que la cantidad de números a sumar no exceda n
Paso 2.2: Ingresar el número a sumar
Paso 2.3: Acumular la sumatoria de los números ingresados
Paso 3: Mostrar el total de la suma de los números

Representación

Diagrama de flujo



· **Seudocódigo**

```
/* El programa halla la suma de cualquier cantidad de números */
PROGRAMA Sumatoria
AUTOR Juan Pérez
CÓDIGO 2O20200
FECHA 1998.02.06

PRECONDICIÓN n pertenece a los naturales, n diferente de cero, x pertenece a los reales;

POSTCONDICIÓN y pertenece a los reales;

Inicio

Variables
  n : entero;      /* Cantidad de números */
  x : real;        /* Número a sumar */
  c : entero;      /* Contador */
  y : real;        /* Suma de los n números */

/* Paso 1: Ingresar la cantidad de números a sumar (verificando la precondición n pertenece a los naturales y n diferente de cero) */
Haga
        Leer(n);
Mientras_que(n<=0)

/* Paso 2: Calcular la suma de los números */
c:=0;
y:=0;

/* Paso 2.1: Verificar que la cantidad de números a sumar no exceda n */
Mientras_que (c<n) haga
        /*Paso 2.2: Ingresar el número a sumar*/
        Leer(x);
        /*Paso 2.3: Acumular la sumatoria de los números ingresados*/
        y:=y+x;
        c:=c+1;
Fin_Mientras_que

/* Paso 3: Mostrar el total de la suma de los números */
        Escribir (y);

Fin
```

Prueba de escritorio

	Variables			
Corrida	*n*	*x*	*c*	*y*
1	*-5*	*Se vuelve a leer la variable n, pues -5 no pertenece a los naturales*		
2	*0*	*Se vuelve a leer la variable n, pues n no puede ser cero porque la sumatoria de cero no es conocida*		
3	*3*	*3*	*0*	*0*
		2	*1*	*3*
			2	*5*

3.3 IMPLEMENTACIÓN DE LA SOLUCIÓN

Codificación

```
/* El programa halla la suma de cualquier cantidad de números */
PROGRAMA Sumatoria
AUTOR Juan Pérez
CÓDIGO 2O20200
FECHA 1998\02\06

PRECONDICIÓN n pertenece a los naturales, n diferente de cero, x pertenece a los reales, y pertenece a los reales;

POSTCONDICIÓN y pertenece a los reales; */

#include <stdio.h>
#include <conio.h>

void main ( )
{
        /* Variables */
        int n;    /* Cantidad de números */
        float x;  /* Número a sumar */
        int c;    /* Contador */
        float y;  /* Suma de los n números */

/* Paso 1: Ingresar la cantidad de números a sumar (verificando la precondición n pertenece a los naturales y n diferente de cero) */
do
{
  printf("Digite cuantos numeros va a sumar: ");
  scanf("%d",&n);
}while(n<=0);
```

```
/* Paso 2: Calcular la suma de los números */
c=0;
y=0;

/* Paso 2.1: Verificar que la cantidad de números
a sumar no exceda n */
while (c<n)
{
  /*Paso 2.2: Ingresar el número a sumar*/
  printf("NUMERO  %d: ", c);
  printf("\nDigite el numero a sumar <Enter>: ");
  scanf("%f",&x);

  /* Paso 2.3: Acumular la sumatoria de los
  números ingresados */
   y=y+x;
   c=c+1;
}

/* Paso 3: Mostrar el total de la suma de los nú-
meros */
  printf("\nLa suma de los n numeros es: %f ",y);
  getch( );
}
```

BIBLIOGRAFÍA

DENNING, P.; Comer, D.; Mulder, M.; Tucker, A.; Turner, A,, y Young, P. "Computing as a Discipline", Report of the ACM Task Force on the Core of Computer Science. En: Computer, February, 1989.

GIBBS, Wait. La crisis crónica de la programación. En: Investigación y Ciencia, Noviembre, 1994.

JOYANES Aguilar, Luis. Fundamentos de programación: Algoritmos y Estructura de Datos. Colombia: McGraw-Hill, 1997.

KNUTH, Donald E. Fundamental Algorithms. Massachusetts: Addison - Wesley, 1973.

MAYER, Richard E. Pensamiento, Solución de Problemas y Cognición. 1a.ed. Barcelona: Ediciones Paidos, 1986.

POLYA, G. Cómo plantear y resolver problemas: un nuevo enfoque del método matemático. México: Editorial Trillas, 1965.

TUCKER, Allen B.; Cupper, Robert D.; Bradley,W. James; Garnick, David K. Fundamentos de Informática: Lógica, solución de problemas, programas y computadoras. México: McGraw Hill, 1994.