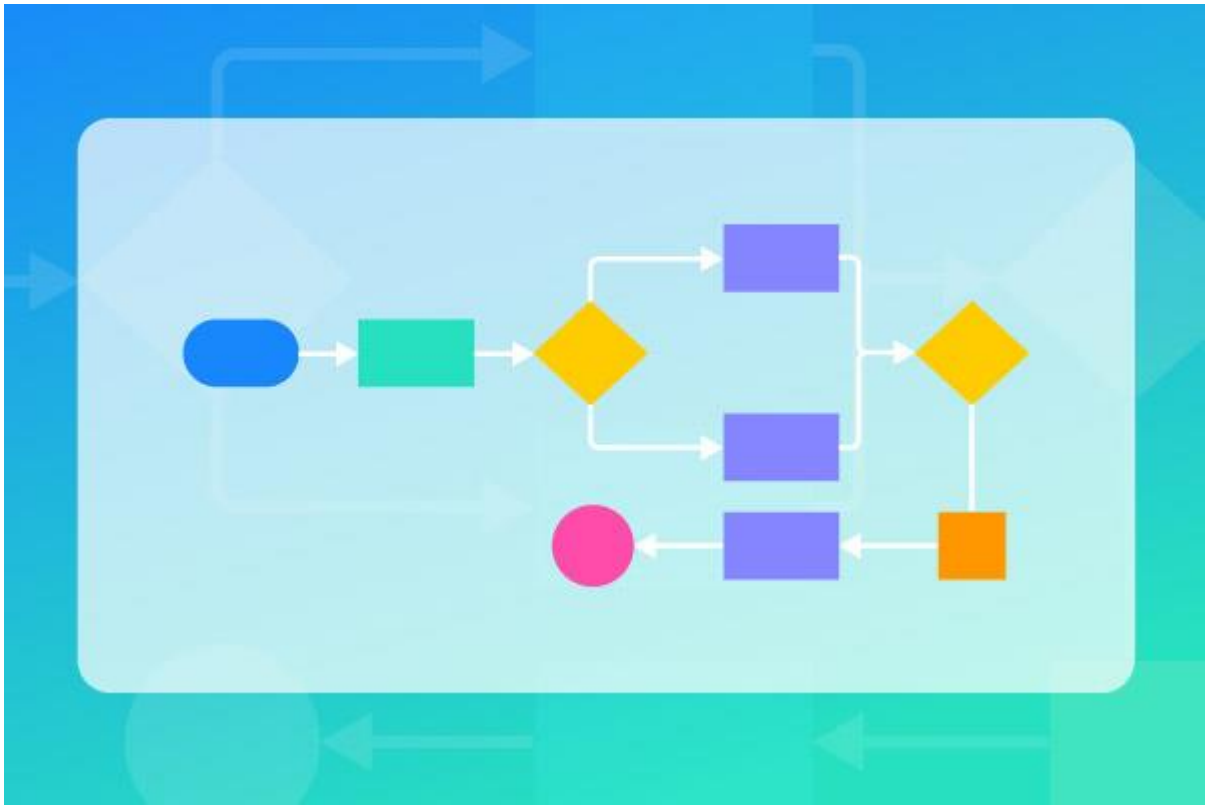
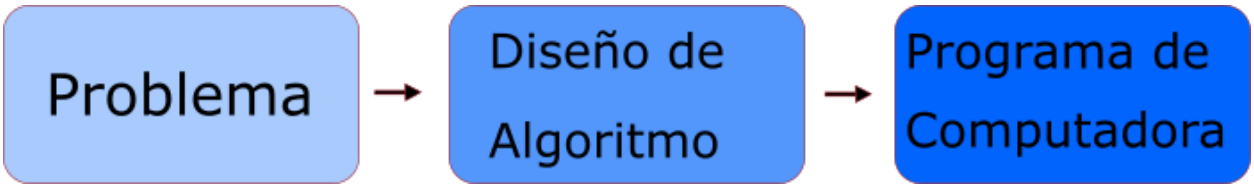


INTRODUCCION A LA ISC

1 Conceptos básicos y metodología para la solución de problemas por medio de equipos informáticos.

El objetivo principal de la materia es el de enseñar a resolver problemas mediante una computadora. Un programador de computadoras antes que nada es un solucionador de problemas.

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



1.1 Introducción a los Algoritmos

Los algoritmos poseen hoy una gran importancia tanto para informática, matemática, robótica y ciencias de la computación, por medio de algoritmos se llega a un orden de ideas y un proceso correcto en la resolución de problemas, elaboración de maquinarias y robots lo que conlleva a un avance en la tecnología.

1.2 Definición de Algoritmo

El algoritmo es una serie de operaciones detalladas
y no ambiguas para ejecutar paso a paso que conducen a la resolución de un problema, y se representan mediante una herramienta o técnica.

Características De Los Algoritmos.
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.

1.3 Tipos de Algoritmos

Los algoritmos se definen como diseños rigurosos y lógicos. Según su clasificación podemos encontrar un tipo para cada área, tarea o trabajo

1.3.1 Cualitativos

Son aquellos en los que se describen los pasos de forma Narrada.

Algoritmo para hacer mi agenda diaria.

INICIO

Levantarme a las 6:00 am y bañarme
Dirigirme al trabajo a las 6:50
Entrar al trabajo a las 7:00 pm
Salir del trabajo a las 3:00 pm
Adelantar tareas de 3:30 a 4:00 pm
Tomar el baño
Ir a la escuela de 5:30 a 10:15
Cenar a las 10:40
Dormir a las 3:00 am

FIN

1.3.2 Cuantitativos

Son aquellos en los que se utilizan cálculos numéricos para definir los pasos del proceso.

Obtener la suma de 2 números.

1. Inicio
2. Declarar (a,b,c)
3. Ingresar (a,b)
4. c=a+b
5. Mostrar (c)
6. Fin

1.4 Definición de Lenguaje

Es una serie de símbolos y reglas que se utilizan para describir de manera explícita un proceso.

1.5 Tipos de Lenguajes Algorítmicos

Las herramientas o técnicas de programación que más se utilizan y que se emplearán para la representación de algoritmos.

1.5.1 Gráficos

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

1.5.2 No Gráficos

Representa en forma descriptiva las operaciones que debe realizar un algoritmo (pseudocodigo).

1.6 Metodología para la solución de problemas por medio de computadora

El proceso de resolución de un problema con una computadora conduce a la escritura de un programa y a su ejecución en la misma. Aunque el proceso de diseñar programas es, esencialmente, un proceso creativo, se puede considerar una serie de fases o pasos comunes, que generalmente deben seguir todos los programadores.

1.6.1 Definición del problema

Identificas cual es el problema.

1.6.2 Análisis del problema

Esta fase requiere una clara definición, donde se contemple exactamente lo que debe hacer el programa y el resultado o solución deseada.

1.6.3 Diseño del algoritmo

En la etapa de análisis del proceso de programación se determina qué hace el programa. En la etapa de diseño se determina cómo hace el programa la tarea solicitada. Los métodos más eficaces para el proceso de diseño se basan en el conocido divide y vencerás. Es decir, la resolución de un problema complejo se realiza dividiendo el problema en subproblemas y a continuación dividiendo estos subproblemas en otros de nivel más bajo, hasta que pueda ser implementada una solución en la computadora.

1.6.4 Codificación

Se implementa el algoritmo en un código escrito en un lenguaje de programación, que refleja las ideas desarrolladas en las fases de análisis y diseño.

La compilación tiene como resultado el código máquina, que es la traducción del código fuente, mediante el empleo de intérpretes o compiladores. A continuación, el programa compilado es alimentado en la computadora para que ésta lo ejecute.

1.6.4.1 Prueba y Depuración

La verificación y la depuración son procesos sucesivos mediante los que se comprueba un programa con una amplia variedad de datos de entrada (“datos de prueba”), para determinar si hay errores, identificarlos y corregirlos.

1.6.5 Documentación

La documentación de un programa se clasifica en interna y externa. La primera se incluye en el código del programa mediante comentarios que ayudan a comprenderlo. Para funcionar el programa no necesita la existencia de comentarios; más aún, los comentarios no generan código ejecutable cuando se traducen a código de máquina. Su única finalidad es hacer que los programas sean más fáciles de leer y comprender, sobre todo para aquellas personas ajenas a ellos, e incluso para los propios programadores.

La documentación externa, por su parte, debe incluir:
1. Listado del programa fuente, en el que se incluyan mapas de memoria, referencias cruzadas y cualquier otra cosa que se obtenga en el proceso de compilación.
2. Explicación de cualquier fórmula o cálculos y expresiones complejas en el programa.
3. Especificación de datos y formatos de pantalla para su entrada y salida, así como cuantas consideraciones se estimen oportunas para mejorar la eficiencia del programa.

En general, la documentación externa se compone de un manual de instalación, un manual de usuario y un manual de mantenimiento

1.6.6 Entrega y Mantenimiento

En ingeniería del software, el mantenimiento de software es la modificación de un producto de software después de la entrega, para corregir errores, mejorar el rendimiento, u otros

atributos.1 El mantenimiento del software es una de las actividades más comunes en la ingeniería de software.

2 Entidades Primitivas para el desarrollo de Algoritmos

2.1 Definición de Dato

El primer objetivo de toda computadora es el manejo de datos. Estos datos pueden ser las cifras de ventas de un supermercado o las calificaciones de una clase. Un dato es la expresión general que describe los objetos con los cuales opera una computadora.

2.2 Tipos de Datos

Son el conjunto básico de tipos provistos por el lenguaje y que, eventualmente, permiten construir tipos de datos más complejos.

2.2.1 Simples

2.2.1.1 Numéricos

Permiten representar valores escalares de forma numérica, esto incluye a los números enteros y los reales. Este tipo de datos permite realizar operaciones aritméticas comunes.

2.2.1.2 Lógicos (booleanos)

Son aquellos que solo pueden tener dos valores (verdadero o falso) ya que representan el resultado de una comparación entre otros datos.

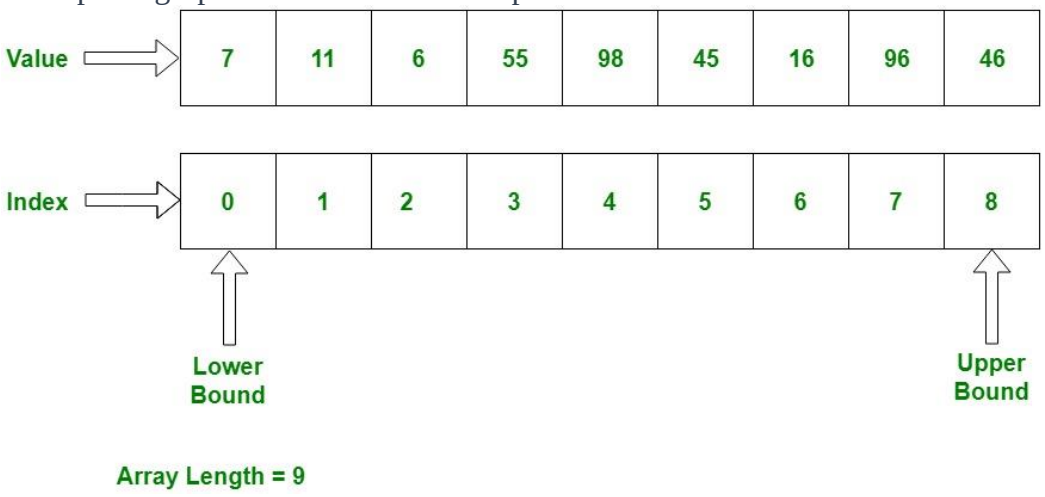
2.2.1.3 Alfanuméricos

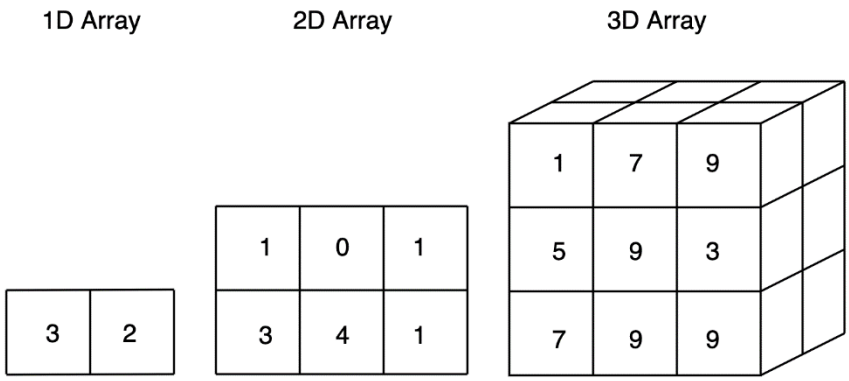
Es una secuencia de caracteres alfanuméricos que permiten representar valores identificables de forma descriptiva, esto incluye nombres de personas, direcciones. Este tipo de datos se representa entre comillas.

2.2.2 Estructurados

2.2.2.1 Arreglos

Sirve para agrupar datos de un mismo tipo con un único nombre.'





2.2.2.2 Registros

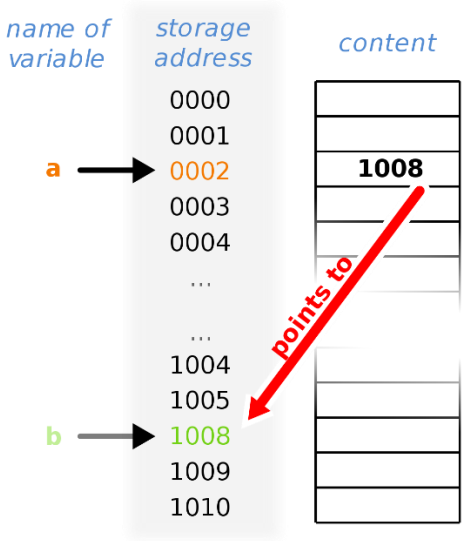
Un registro, en programación, es un tipo de dato estructurado formado por la unión de varios elementos bajo una misma estructura. Estos elementos pueden ser, o bien datos elementales (entero, real, carácter), o bien otras estructuras de datos. A cada uno de esos elementos se le llama campo.

2.2.2.3 Archivos

Un archivo es una secuencia de elementos del mismo tipo, que residen generalmente en memoria auxiliar. Los archivos son utilizados cuando se desea que los datos puedan recuperarse aún después de haber apagado la máquina y también cuando se manejan grandes volúmenes de información.

2.2.2.4 Punteros

Un puntero o apuntador es una variable que da referencia a una región de memoria.



2.2.3 Identificadores

Los identificadores son los nombres que se les asignan a los objetos, los cuales se pueden considerar como variables o constantes, éstos intervienen en los procesos que se realizan para la solución de un problema, por consiguiente, es necesario establecer qué características tienen.

3 Constantes, Variables y Operadores

3.1 Definición de Constante

Un identificador se clasifica como constante cuando el valor que se le asigna a este identificador no cambia durante la ejecución o proceso de solución del problema.

3.2 Definición de Variable

Los identificadores de tipo variable son todos aquellos objetos cuyo valor cambia durante la ejecución o proceso de solución del problema.

3.3 Clasificación de Variables

Los elementos que cambian durante la solución de un problema se denominan variables, se clasifican dependiendo de lo que deben representar en el algoritmo.

3.3.1 Por su Contenido

3.3.1.1 Numéricas

Son aquellas en las cuales se almacenan valores numéricos, positivos o negativos, es decir almacenan números del 0 al 9, signos (+ y -) y el punto decimal.

3.3.1.2 Lógicas

Son aquellas que solo pueden tener dos valores (cierto o falso) estos representan el resultado de una comparación entre otros datos.

3.3.1.3 Alfanuméricas

Está formada por caracteres alfanuméricos (letras, números y caracteres especiales).

3.3.2 Por su Uso

3.3.2.1 De Trabajo

Reciben el resultado de una operación matemática completa y que se usan normalmente dentro de un programa.

3.3.2.2 Contadores

Se utilizan para llevar el control del numero de ocasiones en que se realiza una operación o se cumple una condición.

3.3.2.3 Acumuladores

Forma que toma una variable y que sirve para llevar la suma acumulativa de una serie de valores que se van leyendo o calculando progresivamente.

3.4 Definición de Expresión

Es una combinación de constantes, variables o funciones, que es interpretada de acuerdo a las normas particulares de precedencia y asociación para un lenguaje de programación en particular.

Expresión relacional: y>8

Expresión aritmética: 3+2, x+1

3.5 Definición de Operador

Un operador, es un elemento que indica el tipo de operación que se le va a aplicar a uno o más datos.

3.6 Clasificación de Operadores

3.6.1 Aritméticos

Las expresiones aritméticas son análogas a las fórmulas matemáticas. Las variables y constantes son numéricas (real o entera) y las operaciones son las aritméticas.

+	suma
-	resta
*	multiplicación
/	división
↑, **, ^	exponenciación
mod, %	módulo (resto)

3.6.1.1 Prioridad en Operadores Aritméticos

El orden normal de las operaciones, o de preferencia, es de izquierda a derecha, evaluando en orden los siguientes operadores:

Términos entre paréntesis
Potenciación
Multiplicación y división
Suma y resta.

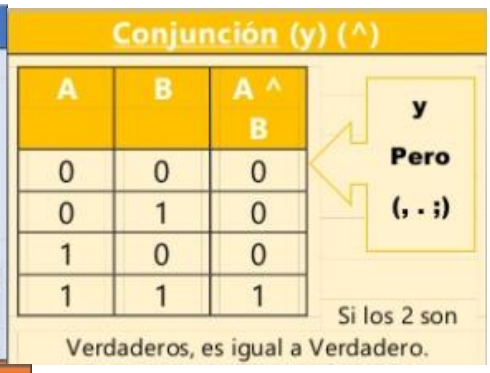
3.6.2 Relacionales

Los operadores relacionales son símbolos que se usan para comparar dos valores. Si el resultado de la comparación es correcto la expresión considerada es verdadera, en caso contrario es falsa.

Operadores Relacionales			
Operador	**Operación**	**Ejemplo**	**Resultado**
=	Igual que	'hola' = 'lola'	FALSO
<>	Diferente a	'a' <> 'b'	VERDADERO
<	Menor que	7 < 15	VERDADERO
>	Mayor que	22 > 11	VERDADERO
<=	Menor o igual que	15 <= 22	VERDADERO
>=	Mayor o igual que	35 > = 20	VERDADERO

3.6.3 Lógicos

Mientras que los operadores aritméticos se usan principalmente con números, los operadores lógicos están pensados para usarse con valores lógicos (verdadero y falso). Hay solo tres operadores lógicos: y, o y no.



4 Diseño de Algoritmos

4.1 Tipos de Diseño de Algoritmos

4.1.1 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 a 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.

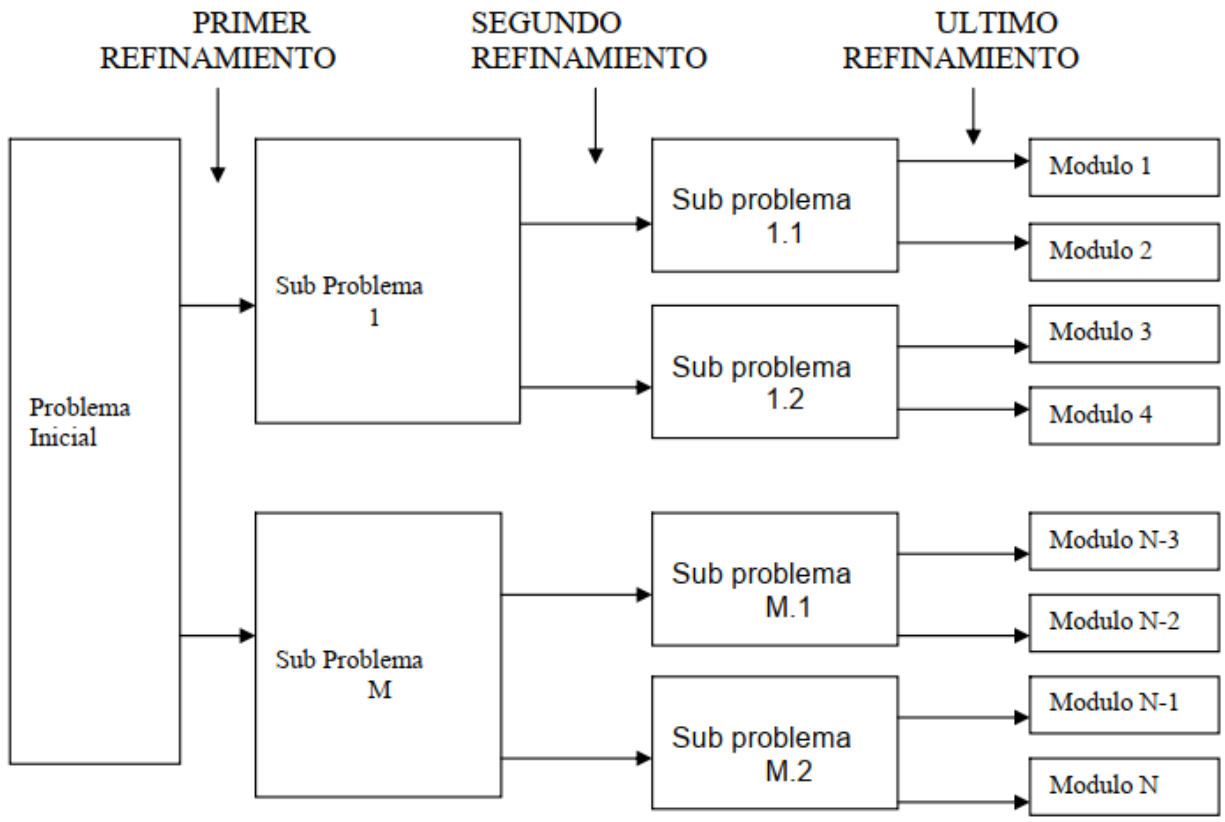
Este diseño consiste en una serie de descomposiciones sucesivas del problema inicial, que recibe el refinamiento progresivo del repertorio de instrucciones que van a formar parte del programa.

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.



4.1.2 Bottom Up

El diseño ascendente se refiere a la identificación de aquellos procesos que necesitan computarizarse con forme vayan apareciendo, su análisis como sistema y su codificación, o bien, la adquisición de paquetes de software para satisfacer el problema inmediato.

Cuando la programación se realiza internamente y haciendo un enfoque ascendente, es difícil llegar a integrar los subsistemas al grado tal de que el desempeño global, sea fluido.

Los problemas de integración entre los subsistemas son sumamente costosos y muchos de ellos no se solucionan hasta que la programación alcanza la fecha límite para la integración total, del sistema. En esta fecha, ya se cuenta con muy poco tiempo, presupuesto o paciencia de los usuarios, como para corregir aquellas delicadas interfaces, que, en un principio, se ignoran.

Aunque cada subsistema parece ofrecer lo que se requiere, cuando se contempla al sistema como una entidad global, adolece de ciertas limitaciones por haber tomado un enfoque ascendente. Uno de ellos es la duplicación de esfuerzos para accesar el software y más aun al introducir los datos. Otro es, que se introducen al sistema muchos datos carentes de valor. Un tercero y tal vez el más serio inconveniente del enfoque ascendente, es que los objetivos globales de la organización no fueron considerados y en consecuencia no se satisfacen.

4.2 Escritura de un algoritmo

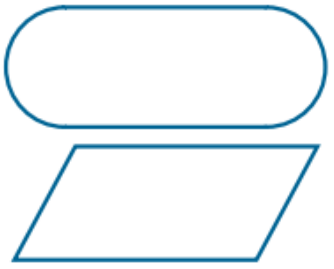
5 Diagramas de Flujo

Los diagramas de flujo son una herramienta que permite representar visualmente qué operaciones se requieren y en qué secuencia se deben efectuar para solucionar un problema dado.

Toda representación gráfica, de cualquier tipo sea, debe cumplir las siguientes cualidades.

- Sencillez. Un método gráfico de diseño de algoritmo debe permitir la construcción de estos de manera fácil y sencilla
- Claridad. Cuando un algoritmo es representado por un método gráfico necesita ser interpretado por otra persona distinta de la que lo diseñó, debe estar lo suficientemente claro para su un fácil reconocimiento de todos los elementos.
- Normalización. Tanto los diseñadores de programas como los usuarios que necesitan la documentación de estos deben utilizar las mismas normas de documentación.
- Flexibilidad. Todo método gráfico de representación debe permitir, sin grandes dificultades, posteriores modificaciones de algunas partes de un algoritmo y la inserción de alguna nueva.

5.1 Simbología de un Diagrama de Flujo



Terminal /Inicio.

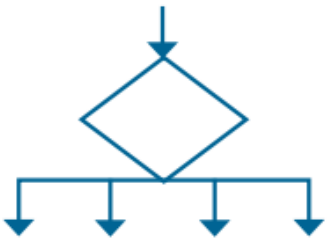
Entrada de datos.

Proceso.

Símbolo de proceso y nos indica la asignación de un valor en la memoria y/o la ejecución de una operación aritmética.

Decisión.

Símbolo de decisión indica la realización de una comparación de valores.



Decisión múltiple.

Imprimir resultados.

Flujo de datos.

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

Conectores.

Conector dentro de página. Representa la continuidad del diagrama dentro de la misma pagina.

Conector de página. Representa la continuidad del diagrama en otra página.

5.2 Recomendaciones para el Diseño de un Diagrama de Flujo

Se deben se usar solamente líneas de flujo horizontales y/o verticales.

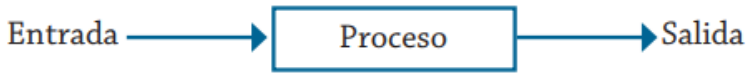
Se debe evitar el cruce de líneas utilizando los conectores.
Se deben usar conectores solo cuando sea necesario.
No deben quedar líneas de flujo sin conectar.
Se deben trazar los símbolos de manera que se puedan leer de arriba hacia abajo y de izquierda a derecha.
Todo texto escrito dentro de un símbolo deberá ser escrito claramente, evitando el uso de muchas palabras.

5.3 Estructuras de Diagramas de Flujo

Sin importar qué herramienta o técnica se utilice para la solución de un problema dado, ésta tendrá una estructura, que se refiere a la secuencia en que se realizan las operaciones o acciones para resolver el problema; esas estructuras pueden ser: secuenciales, de decisión y de ciclo o repetición.

5.3.1 Secuenciales

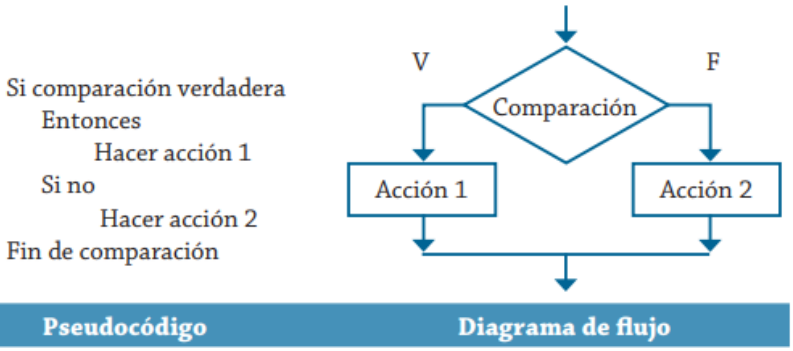
En este tipo de estructura las instrucciones se realizan o se ejecutan una después de la otra y, por lo general, se espera que se proporcione uno o varios datos, los cuales son asignados a variables para que con ellos se produzcan los resultados que representen la solución del problema que se planteó.



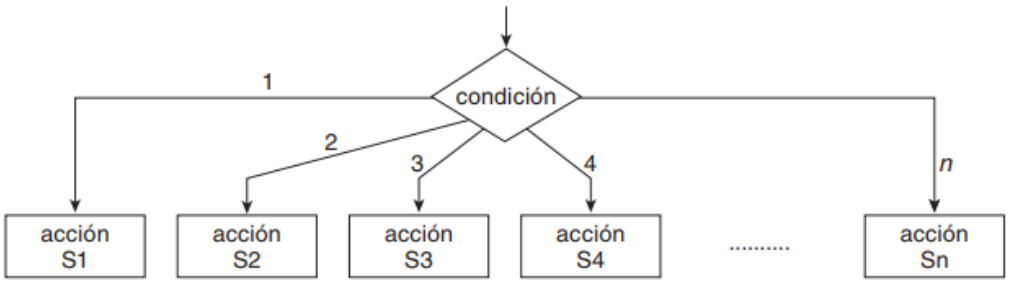
5.3.2 Selección, Bifurcación o Alternativa

En este tipo de estructura las instrucciones se realizan o se ejecutan una después de la otra y, por lo general, se espera que se proporcione uno o varios datos, los cuales son asignados a variables para que con ellos se produzcan los resultados que representen la solución del problema que se planteó.

5.3.2.1 Simple



5.3.2.2 Múltiple



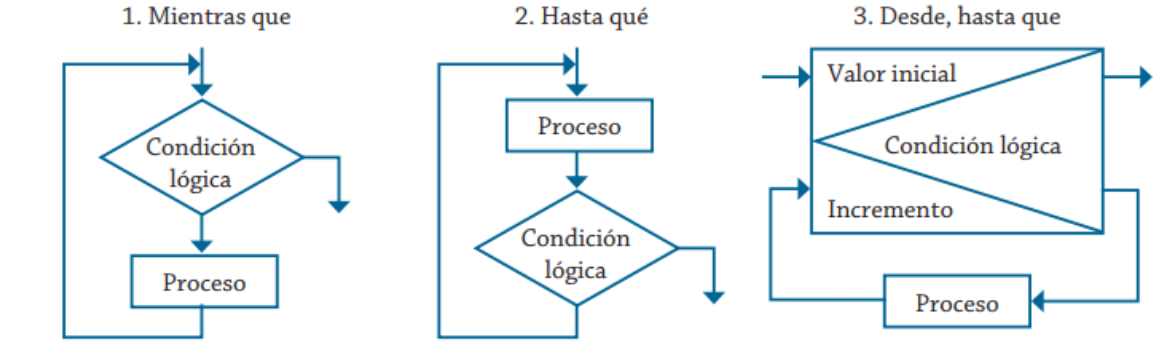
5.3.3 Iteración, Bucle o Repetición

Cuando se requiere que un proceso se efectúe de manera cíclica, se emplean estructuras que permiten el control de ciclos, esas estructuras se emplean con base en las condiciones propias de cada problema, los nombres con los que se conocen éstas son: “Mientras que”, “Repite hasta que” y “Desde, hasta que”.

Para el caso de la estructura “Mientras que”, el ciclo se repite hasta que la condición lógica resulta ser falsa; en tanto que en la estructura “Hasta que”, el ciclo se repite siempre y cuando el resultado de la condición lógica sea falso.

En la estructura “Mientras que” primero se evalúa y luego se realiza el proceso; y para el caso de “Hasta que”, primero se realiza el proceso y luego se evalúa, por consiguiente, este tipo de estructura siempre realizará por lo menos un proceso.

Las estructuras de tipo “Desde” se aplican cuando se tiene definido el número de veces que se realizará el proceso dentro del ciclo, lo que la hace diferente de las otras es que aquellas se pueden utilizar hasta que las condiciones cambien dentro del mismo ciclo, estas condiciones pueden deberse a un dato proporcionado desde el exterior, o bien, al resultado de un proceso ejecutado dentro del mismo, el cual marca el final.



B. Diagrama de flujo

Mientras Condición lógica	**Repite**	**Desde** valor inicial **Hasta** valor Final
Proceso	Proceso	Proceso
Sin Mientras	**Hasta** Condición lógica	**Fin Desde**

A. Pseudocódigo

6 Pseudocódigo

6.1 Definición de Pseudocódigo

El pseudocódigo es una de las herramientas más conocidas para el diseño de solución de problemas por computadora. Esta herramienta permite pasar casi de manera directa la solución del problema a un lenguaje de programación específico. El pseudocódigo es una serie de pasos bien detallados y claros que conducen a la resolución de un problema.

6.2 Estándar de Pseudocódigo

6.3 Palabras Reservadas en Pseudocódigo

Inicio

Leer
Realizar, hacer
Escribir, mostrar
Fin

CRITERIOS Y PROCEDIMIENTOS DE EVALUACIÓN Y ACREDITACIÓN.