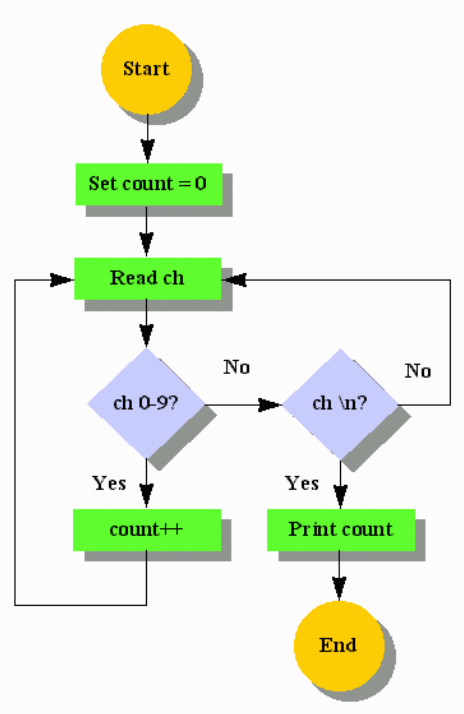
UNIVERSIDAD DISTRITAL
FRANCISCO JOSE DE CALDAS



Introducción a los Algoritmos

2013
Transversal de Programación Básica
Proyecto Curricular de Ingeniería de Sistemas

1. Objetivos

- Asimilar el concepto de Algoritmo y su composición Básica.
- Identificar los componentes de un algoritmo.
- Solucionar problemas utilizando el concepto de algoritmo.
- Solucionar problemas algorítmicamente.

2. Introducción

Diariamente el ser humano trata de dar solución a cada problema que se le presenta; algunas veces la solución de éstos se obtiene al seguir una serie de acciones de manera ordenada, otras veces la solución depende del estado de ánimo de la persona o de las condiciones de su entorno, mientras que en otras ocasiones no se puede llegar a solución alguna.

Esta guía está dividida en dos secciones; la primera sección introduce el concepto de algoritmo, el cual es la base fundamental de la programación de computadores, y la segunda, presenta el concepto de problema para clarificar los tipos de problemas que se pueden solucionar mediante algoritmos.

3. Algoritmo

La palabra algoritmo toma su nombre de Al-khôwarizmi, matemático y astrónomo del siglo IX quien al escribir un tratado sobre manipulación de números y ecuaciones, el Kitab al-jabr w'almugabala, usó en gran medida la noción de lo que se conoce hoy como algoritmo.

Un ALGORITMO es una secuencia finita "bien definida" de tareas "bien definidas", cada una de las cuales se puede realizar con una cantidad de recursos finitos. Se dice que una tarea esta 'bien definida', si se sabe de manera precisa las acciones requeridas para su realización. Aunque los recursos que debe utilizar cada tarea deben ser finitos estos no están limitados, es decir, si una tarea bien definida requiere una cantidad inmensa (pero finita) de algún recurso para su realización, dicha tarea puede formar parte de un algoritmo. Además, se dice que una secuencia de tareas esta 'bien definida' si se sabe el orden exacto de ejecución de cada una de las mismas.

EJECUTAR un algoritmo es realizar las tareas del mismo, en el orden especificado y utilizando los recursos disponibles.

El concepto de algoritmo se ilustra frecuentemente comparándolo con una receta: al igual que las recetas, los algoritmos habitualmente están formados por secuencias de instrucciones que probablemente se repiten (iteran) o que requieren decisiones (comparaciones lógicas) hasta que completan su tarea. Un algoritmo puede no ser correcto, con lo cual, por más que sus pasos se lleven a cabo correctamente, el estado final no será el esperado.

Normalmente, cuando un algoritmo está asociado con el procesamiento de información, se leen datos de una fuente o dispositivo de entrada, se procesan y se emiten por un dispositivo de salida, o bien se almacenan para su uso posterior. Los datos almacenados se consideran parte del estado interno de la entidad que ejecuta el algoritmo.

Dado que un algoritmo es una lista precisa de pasos, el orden de ejecución será casi siempre crítico para su funcionamiento. En general, se asume que las instrucciones se enumeran explícitamente, y deben ejecutarse "desde arriba hacia abajo", lo cual se establece más formalmente según el concepto de **flujo de control**. Esta forma de "pensar" el algoritmo asume las premisas del paradigma de programación imperativa. Dicho paradigma es el más común, e intenta describir las tareas en términos "mecánicos" y discretos. Los paradigmas de la programación funcional y de la programación lógica describen el concepto de algoritmo en una forma ligeramente diferente.

Para definirlo en forma matemáticamente precisa, Alan Mathison Turing –famoso matemático inglés (1912-1954), cuyas contribuciones en el campo de la matemática y de la teoría de la computación le han valido ser considerado uno de los padres de la computación digital– ideó un dispositivo imaginario al que denominó **máquina de computación lógica** (LCM, *Logical Computing Machine*), pero que ha recibido en su honor el nombre de **máquina de Turing**. Lo que confiere a este supuesto dispositivo su extraordinaria importancia es que es capaz de resolver cualquier problema matemático, a condición de que el mismo haya sido reducido a un algoritmo. Por este motivo, se considera que algoritmo es cualquier conjunto de operaciones que pueda ser ejecutado por la máquina de Turing.

3.1 Características de un algoritmo

- Son independientes del lenguaje de programación a u utilizar.
- Sencillo, los pasos deben ser claros y bien definidos.
- Precisos, indican claramente el orden de realización paso a paso.
- Definidos, cada vez que se ejecutan con las mismas entradas se obtiene el mismo resultado.
- Finitos, tienen un número de pasos finito.

Precisión	Definitud o Determinismo	Finitud
El algoritmo debe indicar el orden exacto de ejecución de cada tarea.	Si se sigue el algoritmo dos o más veces con los mismos datos de entrada, se deben obtener los mismos datos de salida.	El algoritmo debe terminar en algún momento y debe usar una cantidad de recursos finita.

Dada una cantidad de datos de entrada de un algoritmo, se dice que la cantidad de un recurso usada por dicho algoritmo para su ejecución determina la complejidad del algoritmo respecto a tal recurso. Cuando se implementa un algoritmo en un computador digital, los recursos con los que se

cuenta son tiempo de proceso y memoria. Por lo tanto, a un algoritmo implementado en un computador digital se le pueden calcular sus complejidades temporal y espacial.

3.2 Estructura básica de un algoritmo

Un algoritmo se constituye por tres elementos:

Datos	Instrucciones	Estructuras de control
Lo que el algoritmo recibe, procesa y entrega como resultado.	Las acciones o procesos que el algoritmo realiza sobre los datos.	Las que determinan el orden en que se ejecutarán las instrucciones del algoritmo.

3.3 EJEMPLOS DE ALGORITMOS

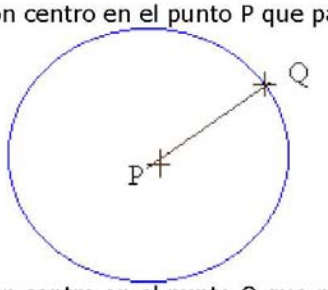
Problema No. 1: Un estudiante se encuentra en su casa (durmiendo) y debe ir a la universidad (a tomar la clase de programación!!), ¿qué debe hacer el estudiante?

Inicio
PASO 1. Dormir
PASO 2. **Hacer** 1 **hasta** que suene el despertador (o lo llame la mamá).
PASO 3. Mirar la hora.
PASO 4. ¿Hay tiempo suficiente?
PASO 4.1. Si hay, **entonces**
PASO 4.1.1. Bañarse.
PASO 4.1.2. Vestirse.
PASO 4.1.3. Desayunar.
PASO 4.2. Sino,
PASO 4.2.1. Vestirse.
PASO 5. Cepillarse los dientes.
PASO 6. Despedirse de la mamá y el papá.
PASO 7. ¿Hay tiempo suficiente?
PASO 7.1. Si hay, **entonces**
PASO 7.1.1. Caminar al paradero.
PASO 7.2. Sino, Correr al paradero.
PASO 8. Hasta que pase un bus para la universidad **hacer**:
PASO 8.1. Esperar el bus
PASO 8.2. Ver a las demás personas que esperan un bus.
PASO 9. Tomar el bus.
PASO 10. Mientras no llegue a la universidad **hacer**:
PASO 10.1. Seguir en el bus.
PASO 10.2. Pelear mentalmente con el conductor.
PASO 11. Timbrar.
PASO 12. Bajarse.
PASO 13. Entrar a la universidad.
Fin

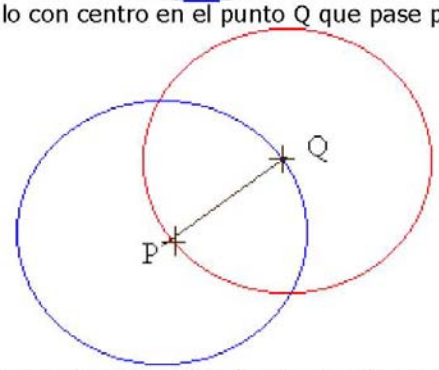
Problema No. 2: Sean los puntos $P=(a,b)$ y $Q=(c,d)$ que definen una recta, encontrar un segmento de recta perpendicular a la anterior que pasa por el punto medio de los puntos dados.

Inicio

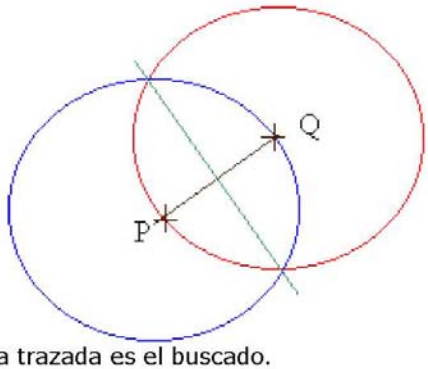
PASO 1. Trazar un círculo con centro en el punto P que pase por el punto Q.



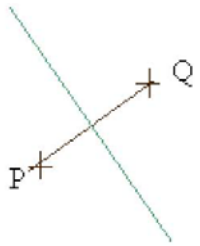
PASO 2. Trazar un círculo con centro en el punto Q que pase por el punto P.



PASO 3. Trazar un segmento de recta entre los puntos de intersección de las circunferencias trazadas.



Fin. El segmento de recta trazada es el buscado.



Problema No. 3: Realizar la suma de los números 2448 y 5746.

Inicio
PASO 1. Colocar los números el primero encima del segundo, de tal manera que las unidades, decenas, centenas, etc., de los números queden alineadas. Trazar una línea debajo del segundo número.
PASO 2. Empezar por la columna más a la derecha.
PASO 3. Sumar los dígitos de dicha columna.
PASO 4. Si la suma es mayor a 9 anotar un 1 encima de la siguiente columna a la izquierda y anotar debajo de la línea las unidades de la suma. Si no es mayor anotar la suma debajo de la línea.
PASO 5. Si hay más columnas a la izquierda, pasar a la siguiente columna a la izquierda y volver a 3.
PASO 6. El número debajo de la línea es la solución.
Fin

Problema No. 4: Cambiar la rueda pinchada de un automóvil teniendo un gato mecánico en buen estado, una rueda de reemplazo y una llave inglesa.

Inicio
PASO 1. Aflojar los tornillos de la rueda pinchada con la llave inglesa.
PASO 2. Ubicar el gato mecánico en su sitio.
PASO 3. Levantar el gato hasta que la rueda pinchada pueda girar libremente.
PASO 4. Quitar los tornillos y la rueda pinchada.
PASO 5. Poner rueda de repuesto y los tornillos.
PASO 6. Bajar el gato hasta que se pueda liberar.
PASO 7. Sacar el gato de su sitio.
PASO 8. Apretar los tornillos con la llave inglesa.
Fin

Problema No. 5: Encontrar los números primos entre 1 y 50.

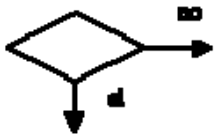
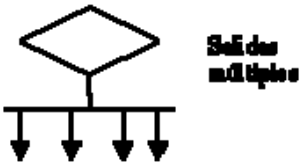
Inicio
PASO 1. Escribir los números de 1 al 50
PASO 2. Tachar el número 1 ya que no es primo.
PASO 3. Para k entre 2 y el entero más cercano por debajo de la raíz cuadrada de 50, si el número k no esta tachado, tachar los múltiplos del número k, sin tachar el número k.
PASO 4. Los números que no se tacharon son los números primos entre 1 y 50.
Fin

3.4 REPRESENTACIÓN DE LOS ALGORITMOS

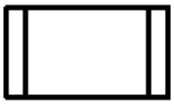
Las técnicas para la representación de algoritmos utilizadas más comúnmente son:

1. **Diagramas de Flujo**: Se basan en la utilización de diversos símbolos para representar operaciones específicas. Se les llama diagramas de flujo porque los símbolos utilizados se conectan por medio de flechas para indicar la secuencia de operación.

SIMBOLOGIA UTILIZADA EN LOS DIAGRAMAS DE FLUJO

SIMBOLO	FUNCION
	Terminal (representa el Inicio y el Final, de un programa, puede representar también una pausa o interrupción programada que sea necesario realizar en un programa.
	Entrada/Salida (cualquier tipo de introducción de datos)
	Proceso (cualquier tipo de operación que pueda originar cambio de valor, formato o posición de la información almacenada en memoria, operaciones aritméticas).
	Decisión (indica operaciones lógicas o de comparación entre datos -normalmente dos- y en función del resultado de la misma determina cual de los distintos caminos alternativos del programa se debe seguir).
	Decisión múltiple (en función del resultado de la comparación se seguirá uno de los diferentes caminos de acuerdo con dicho resultado).
	Conector (Sirve para enlazar dos partes cualesquiera de un organigrama a través de un conector en la salida y otro conector en la salida)
	Indicador de dirección o línea de flujo (indica el sentido de ejecución de las operaciones)
	Línea conectora (sirve de unión entre dos símbolos).

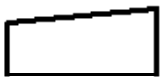
Conector (Conexión entre dos puntos del organigrama situado en páginas diferentes).



Llamada a subrutina o a un proceso predeterminado (una subrutina es un módulo independiente del programa)

Pantalla (se utiliza en ocasiones en lugar del símbolo de E/S).

Impresora (se utiliza en ocasiones en lugar del símbolo de E/S).



Teclado (Se utiliza en ocasiones en lugar del símbolo de E/S).

2. **Pseudocódigo:** Es un lenguaje de especificación de algoritmos. El uso de tal lenguaje hace el paso de codificación final (esto es, la traducción a un lenguaje de programación) relativamente fácil. El pseudocódigo nació como un lenguaje similar al lenguaje natural y era un medio para representar básicamente las estructuras de control de programación estructurada.

   Se considera un primer borrador, dado que el pseudocódigo tiene que traducirse posteriormente a un lenguaje de programación. La ventaja del pseudocódigo es que en su uso en la planificación de un programa, el programador se puede concentrar en la lógica y en las estructuras de control y no preocuparse de las reglas de un lenguaje específico. Es también fácil modificar el pseudocódigo si se descubren errores o anomalías en la lógica del programa, además de todo esto es fácil su traducción a lenguajes como *Pascal*, *C* o *Basic.* El pseudocódigo utiliza para representar las acciones sucesivas palabras reservadas (similares a sus homónimos en los lenguajes de programación), tales como inicio, fin, si-entonces-sino, mientras,….etc.

Secuencia	Decisión Simple	Decisión Doble
Inicio **acción1** **acción2** **.**	**si condición entonces** **acción1** **acción2** **.**	**si condición entonces** **acción1** **acción2** **.**

. . **acción n** **Fin**	. . **Acción n**	. . **en caso contrario** **acción1** **acción2**

Iteración Fija	Condicional al inicio	Condicional al final
para var. Entera inicial **hasta final hacer** **acción1** **acción2** . . . **Acción n**	**mientras condición hacer** **acción1** **acción2** . . . **Acción n**	**repita** **acción1** **acción2** . . . **acción n** **Hasta que condición**

Selección **casos selector de** **valor1 : acción1** **acción2** **valor2 : acción1** **acción2** ... **valor n : acción1** **acción2**		

Nomenclatura:

```
<variable>
[opcional]
palabra_reservada
/*comentario*/
```

Forma General de un Programa

```
[<definición de registros o tipos de datos>]
[constantes <declaración constantes>]
[variables <declaración variables globales del
programador>]
[<definición de funciones y procedimientos>]

procedimiento principal()
[constantes <declaración constantes>]
[variables <declaración variables>]
inicio
   /*bloque de instrucciones*/
fin_procedimiento
```

Tipos de datos primitivos:

<tipo>
entero
real
caracter
booleano

4. LOS PROBLEMAS

Se tiene un problema cuando se desea encontrar uno o varios objetos desconocidos (ya sean estos números, diagramas, figuras, demostraciones, decisiones, posiciones, algoritmos, u otras cosas), que cumplen condiciones y/o relaciones, previamente definidas, respecto a uno o varios objetos conocidos. De esta manera, solucionar un problema es encontrar los objetos desconocidos de dicho problema.

4.1 CLASIFICACIÓN DE PROBLEMAS

Los problemas se clasifican por la existencia de una solución en solubles, no solubles e indecidible.

- Un problema se dice SOLUBLE si se sabe de antemano que existe una solución para él.
- Un problema se dice INSOLUBLE si se sabe que no existe una solución para él.
- Un problema se dice INDECIDIBLE si no se sabe si existe o no existe solución para él.

A su vez, los problemas solubles se dividen en dos clases: los algorítmicos y los no algorítmicos.

- Un problema se dice ALGORÍTMICO si existe un algoritmo que permita darle solución.
- Un problema se dice NO ALGORÍTMICO si no existe un algoritmo que permita encontrar su solución.

4.2 EJEMPLOS DE PROBLEMAS

1. Sean los puntos P=(a,b) y Q=(c,d) que definen una recta, encontrar un segmento de recta perpendicular a la anterior que pase por el punto medio de los puntos dados

VARIABLES CONOCIDAS	Los puntos *P* y *Q*.
VARIABLES DESCONOCIDAS	Un segmento de recta.
CONDICIONES	El segmento de recta debe pasar por el punto medio entre *P* y *Q*, y debe ser perpendicular a la recta trazada entre *P* y *Q*
TIPO DE PROBLEMA	Soluble-algorítmico. Es soluble por que ya existe un algoritmo que permite encontrar la solución del mismo. Este algoritmo fue presentado en la sección anterior.

2. De las siguientes cuatro imágenes, ¿cuál es la más llamativa?

VARIABLES CONOCIDAS	Las cuatro imágenes.
VARIABLES DESCONOCIDAS	Una de las cuatro imágenes.
CONDICIONES	La imagen buscada es la más llamativa para quien resuelve el problema.
TIPO DE PROBLEMA	Soluble-no algorítmico. La solución de este problema existe, es alguna de las cuatro imágenes presentadas, pero no existe un algoritmo que permita determinar cual es, ya que el concepto de imagen más llamativa no esta bien definido.

3. Un granjero tiene cincuenta animales entre conejos y gansos. Si la cantidad de patas de los animales es ciento cuarenta, ¿cuántos conejos y cuantos gansos tiene el granjero?

VARIABLES CONOCIDAS	La cantidad total de animales, cantidad de patas totales.
VARIABLES DESCONOCIDAS	La cantidad de conejos y la cantidad de gansos.
CONDICIONES	La suma de los conejos y los gansos es igual a cincuenta. La suma de los pies de los conejos (cuatro por cada uno) y de los gansos (dos por cada uno) es igual a ciento cuarenta.
TIPO DE PROBLEMA	Soluble-algorítmico.

4. ¿Existe en la expansión decimal de p una secuencia de tamaño n para cualquier número natural n?

VARIABLES CONOCIDAS	El número n.
VARIABLES DESCONOCIDAS	Un valor de verdad (falso o verdadero).
CONDICIONES	Verdadero si existe en la expansión decimal de p una secuencia de tamaño *n* del número *n*, para todo número natural *n*, Falso en otro caso.
TIPO DE PROBLEMA	Indecidible. Este problema es indecidible por que si en el primer millón de dígitos de p no se encuentra una secuencia como la buscada, nada garantiza que en el siguiente millón de dígitos no se encuentre tal secuencia. Pero si no se encuentra en el segundo millón, nada garantiza que no se encuentre después o no se encuentre. De esta manera no se puede decidir si existe o no existe tal secuencia.

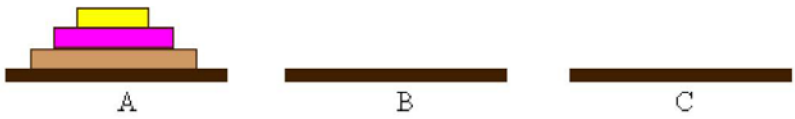
5. Realizar la suma de los siguientes números: 1245893467 y 3464895786

VARIABLES CONOCIDAS	Un número natural.
VARIABLES DESCONOCIDAS	Dos números naturales 1245893467 y 3464895786.
CONDICIONES	El número desconocido es igual a la suma de los dos números dados.
TIPO DE PROBLEMA	Soluble-algorítmico. Es soluble por que ya existe un algoritmo que permite encontrar la solución del mismo. Este algoritmo fue presentado en la sección anterior.

6. Una partícula se mueve en el espacio de manera aleatoria, si en el instante de tiempo t se encuentra en la posición x , ¿cuál será la posición exacta de dicha partícula 10 segundos después?

VARIABLES CONOCIDAS	Posición en el instante de tiempo t.
VARIABLES DESCONOCIDAS	Una posición.
CONDICIONES	La partícula se mueve en el espacio de manera aleatoria.
TIPO DE PROBLEMA	Insoluble. No se puede solucionar por que no existe forma de predecir la posición de la partícula, pues su movimiento es aleatorio, es decir, se mueve de manera arbitraria.

7. Un robot puede apilar (poner encima), en ciertos lugares cajas de diferentes tamaños. La caja a apilar no puede ser más grande que las que ya estén apiladas en dicho lugar. El robot puede solo tomar la caja de más arriba de una pila. Mirando la figura, como puede el robot pasar las tres cajas apiladas en el lugar A, al lugar C usando, si es necesario, el lugar de apilar B.



VARIABLES CONOCIDAS	Número de cajas, posición inicial y posición destino, número de lugares de apilamiento.
VARIABLES DESCONOCIDAS	Una secuencia de apilamientos.
CONDICIONES	Solo se pueden apilar cajas sobre otras más grandes. Solo se puede tomar una caja a la vez y solo la que este más arriba en una pila de cajas.
TIPO DE PROBLEMA	Soluble-algorítmico.

4.3 ESTRATEGIAS PARA LA SOLUCIÓN DE PROBLEMAS

Cuando se trata de resolver un problema, pueden presentarse varias estrategias para solucionarlo. Las técnicas o estrategias más comunes son:

Estrategias de solución directa:

- Algoritmos de Solución Forzada: Un algoritmo de es te tipo resuelve el problema de la forma más simple, obvia o directa. Como resultado es posible que el algoritmo haga más trabajo que una solución más sofisticada. Por otra parte, las soluciones forzadas son más fáciles de implementar y por eso algunas veces resultan más eficientes.
- Algoritmos Codiciosos: Se caracterizan las decisiones que toman se basan en que la búsqueda del menor costo en esa parte del problema, pero no toman en cuenta el resto de la solución y en ocasiones no generan soluciones óptimas.

Estrategias de Vuelta Atrás:

Un algoritmo de vuelta atrás, sistemáticamente considera todos los posibles resultados para cada decisión. En este sentido, los algoritmos vuelta atrás son como las soluciones forzadas. Sin embargo, los algoritmos vuelta atrás se distinguen por la forma en que exploran todas las posibles soluciones; en ocasiones estos algoritmos encuentran que una búsqueda exhaustiva es innecesaria y por lo tanto pueden tener una mejor ejecución.

Estrategias arriba-abajo:

- **Algoritmos divide y vencerás**. Para resolver un problema, este se subdivide en uno o más subproblemas cada uno de los cuales es similar al problema dado. Cada uno de los subproblemas se soluciona en forma independiente y al final las soluciones de todos los subproblemas se combinan para obtener la solución general del problema completo.

Estrategias abajo-arriba

- **Programación Dinámica**: Para resolver un problema se resuelven una serie de subproblemas. La serie de subproblemas es planeada cuidadosamente de tal forma que cada solución subsecuente se obtiene mediante la combinación de las soluciones de uno o más subproblemas que ya han sido resueltos. Todas las soluciones intermedias se mantienen en una tabla para evitar la duplicidad de esfuerzos.

Estrategias Probabilísticas:

- En los algoritmos probabilísticos existe un elemento de aleatoriedad en la forma en que el algoritmo soluciona el problema, se dice que estos métodos son el último recurso debido a que se usan cuando no hay otra técnica conocida que se pueda aplicar. Los métodos probabilísticos se usan cuando el espacio de soluciones es tan grande que una búsqueda exhaustiva no sería factible.

4.4 PROGRAMACIÓN ESTRUCTURADA

La programación estructurada es un estilo de programación en el cual, la estructura de un programa se hace tan clara como sea posible utilizando tres estructuras:

1. Secuencia Simple
2. Selección
3. Iteración

Estos tres tipos de estructuras de control pueden combinarse para producir programas con cualquier tipo de información que se vaya a procesar. Un programa estructurado tiene como característica que puede leerse de arriba hacia abajo lo que hace que el programa sea más fácil de leer y comprender por otros programadores facilitando así su mantenimiento.

Un programa estructurado se compone de segmentos. Cada segmento está constituido por una entrada y una salida, tal segmento de denomina un programa propio.

4.4.1 Teoría de la Programación Estructurada

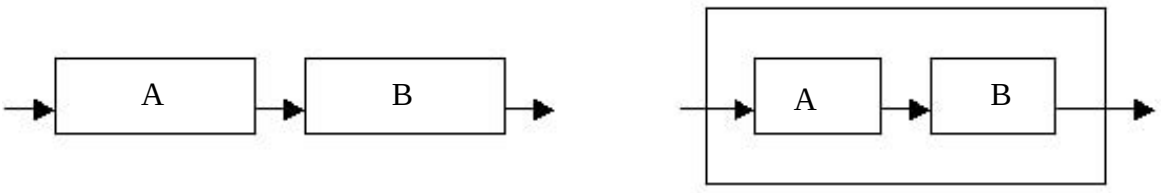
El teorema de la estructura: se refiere a que cualquier programa propio se puede escribir usando solamente las tres estructurad de control: secuencia, selección e iteración

Un programa propio contempla dos segmentos básicos:

a. Tiene exactamente un punto de entrada y uno de salida
b. Dentro de ese punto de entrada y salida hay trayectorias que conducen a cada parte del programa; esto significa que no existen loops infinitos o una codificación inalcanzable.

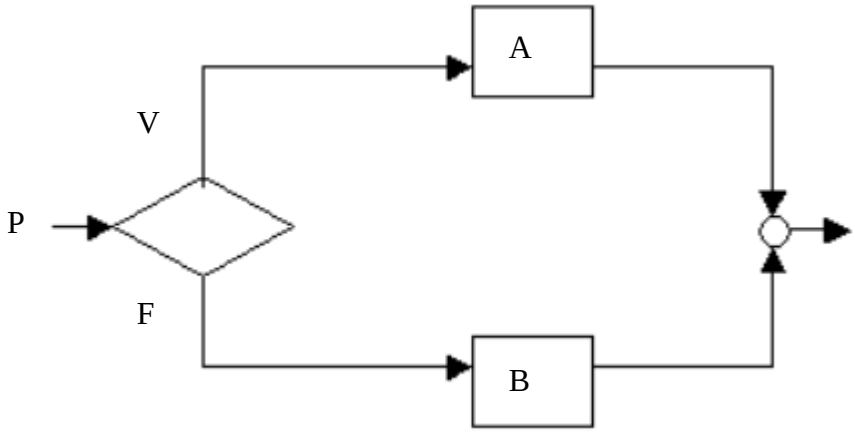
Las tres estructuras de control se ilustran a continuación:

Secuencia: Las instrucciones del programa se ejecutan en el orden en el cual ellas aparecen en el programa como se indica en la siguiente figura:

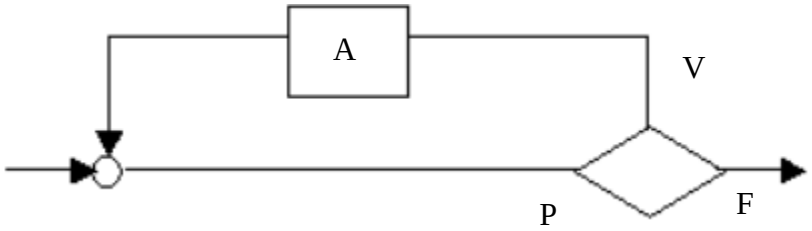


A y B pueden ser simples instrucciones hasta módulos completos. A y B deben ser ambos programas propios en el sentido ya definido de entrada y salida. La combinación de A y B es también un programa propio y que tiene también una entrada y una salida.

Selección: Es escoger entre dos opciones basadas en un predicado. Se conoce como estructura si – entonces – sino P es el predicado y A y B son las afirmaciones.



Iteración: Repetir varias veces una acción hasta cuando deje de cumplirse la condición. Se conoce como la estructura hacer – mientras



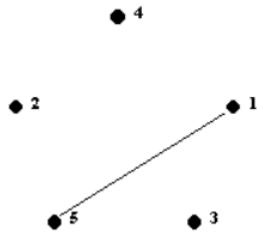
Es de anotar que hay algunas variaciones a esta estructura dependiendo del lenguaje de programación. La idea fundamental es que siempre que aparezca una función que se puede dibujar en recuadro se pueda sustituir por cualquiera de las tres estructuras básicas constituyendo así un programa propio.

5. EJERCICIOS DE PROBLEMAS

Para los siguientes problemas, determine las variables conocidas, desconocidas, las condiciones y el tipo de problema. Para aquellos problemas algorítmicos desarrollar adicionalmente un algoritmo que permita encontrar una solución.

1. Se tienen dos jarras (A y B) de capacidades 3 y 7 litros respectivamente, sobre las cuales se pueden efectuar las siguientes acciones: Llenar totalmente cualquiera de las dos jarras, vaciar una de las dos jarras en la otra hasta que la jarra origen este vacía o hasta que la jarra destino este llena y vaciar el contenido de una jarra (este llena o no) en un sifón. ¿Cómo se puede dejar en la jarra A un solo litro utilizando solamente las anteriores acciones?
2. Es cierta o no es cierta la siguiente frase: “Esta frase no es cierta”.
3. Si Juan tiene el doble de la edad de Pedro y la suma de las edades de los dos es 33 años, ¿Cuántos años tiene Juan y cuántos tiene Pedro?

4. ¿Qué figura se forma al unir los puntos marcados con números consecutivos con una línea?



5. Calcular de manera exacta el número de átomos del universo.
6. Calcular el costo de una serie de productos comprados en el supermercado.
7. Determinar quién es el mejor jugador de fútbol de toda la historia.
8. Construir un barco de papel.

6. EJERCICIOS DE ALGORITMOS

Para los siguientes problemas dar un algoritmo y si es posible una ejecución del mismo.

1. Buscar en el directorio telefónico, el número de:
   a. José González Pérez
   b. Pedro Gómez Bernal.
   c. Escribir un algoritmo que sirva para buscar a cualquier persona.

2. Calcular el número de días entre las fechas:

   a. Enero 17 de 1972 y Julio 20 de 1973
   b. Febrero 2 de 1948 y Agosto 11 de 1966
   c. Escribir un algoritmo que sirva para calcular la cantidad de días entre cualquier dos fechas.

3. Solicitar en préstamo algún libro de una biblioteca.
4. Hacer una caja de cartón con tapa de:
   a. 20 cm de largo, por 10 cm de ancho y 5 cm de alto.
   b. 10 cm de largo, por 30 cm de ancho y 15 cm de alto.
   c. Escribir un algoritmo que sirva para construir una caja de cartón con tapa de cualquier tamaño.

5. Construir un avión de papel.

6. Calcular manualmente la división de cualquier par de números naturales. El resultado también debe ser un número natural. Escribir un algoritmo para calcular el residuo de la división.

7. Un juego muy famoso entre dos niños es el de adivina mi número, el cual consiste en que cada niño trata de adivinar el número pensado por el otro niño. Dicho número generalmente está entre 1 y 100. Las reglas del juego son las siguientes:

   a. Cada niño posee un turno en el que trata de averiguar el número del otro.
   b. En su turno el primer niño pregunta si un número que dice es el pensado por el segundo.
   c. Si el número que ha dicho el primer niño es el que pensó el segundo, este último debe informarle al primero que ganó.
   d. Si el número no es el segundo niño debe decir si su número pensado es menor o mayor al que el primer niño dijo.
   e. Luego el segundo niño tiene su turno y de esta manera se van intercalando hasta que alguno de los dos gane. Desarrollar un algoritmo para jugar adivina mi número.

8. Una balanza se encuentra en equilibrio cuando el producto de la carga aplicada sobre el brazo derecho por la longitud de este brazo, es igual al producto de la carga aplicada sobre el brazo izquierdo por la longitud de este otro brazo. Determinar si la balanza se encuentra en equilibrio si:
   a. La longitud del brazo izquierdo es3 m, la del derecho es 2 m, la carga aplicada al brazo izquierdo es 5 Kg y la carga aplicada al derecho es 7 Kg.
   b. La longitud del brazo izquierdo es4 m, la del derecho es 2 m, la carga aplicada al brazo izquierdo es 4 Kg y la carga aplicada al derecho es 4 Kg.
   c. Desarrollar un algoritmo que sirva para cualquier conjunto de valores para las longitudes de los brazos y las cargas aplicadas.

Lecturas de Profundización:

- El texto fue tomado de: http://200.69.103.48/comunidad/grupos/compuparalela/capitulo2.pdf

Imágenes:
Las imágenes fueron tomadas de www.google.com

Referentes:

- http://www.conocimientosweb.net/portal/article782.html
- http://compinformatidf.files.wordpress.com/2012/03/cap6-algoritmos-cc101.pdf
- http://fcqi.tij.uabc.mx/usuarios/palacios/UNIDAD%201%20%20Introduccion%20a%20los%20algoritmos%20y%20Estructuras%20de%20Datos.pdf
- http://www.virtual.unal.edu.co/cursos/ingenieria/2001839/modulo1/cap_02/leccion_2.htm