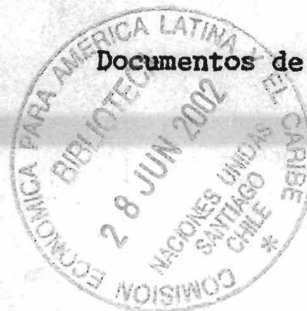


03218 (HP6) c-2

Centro Latinoamericano de Demografía



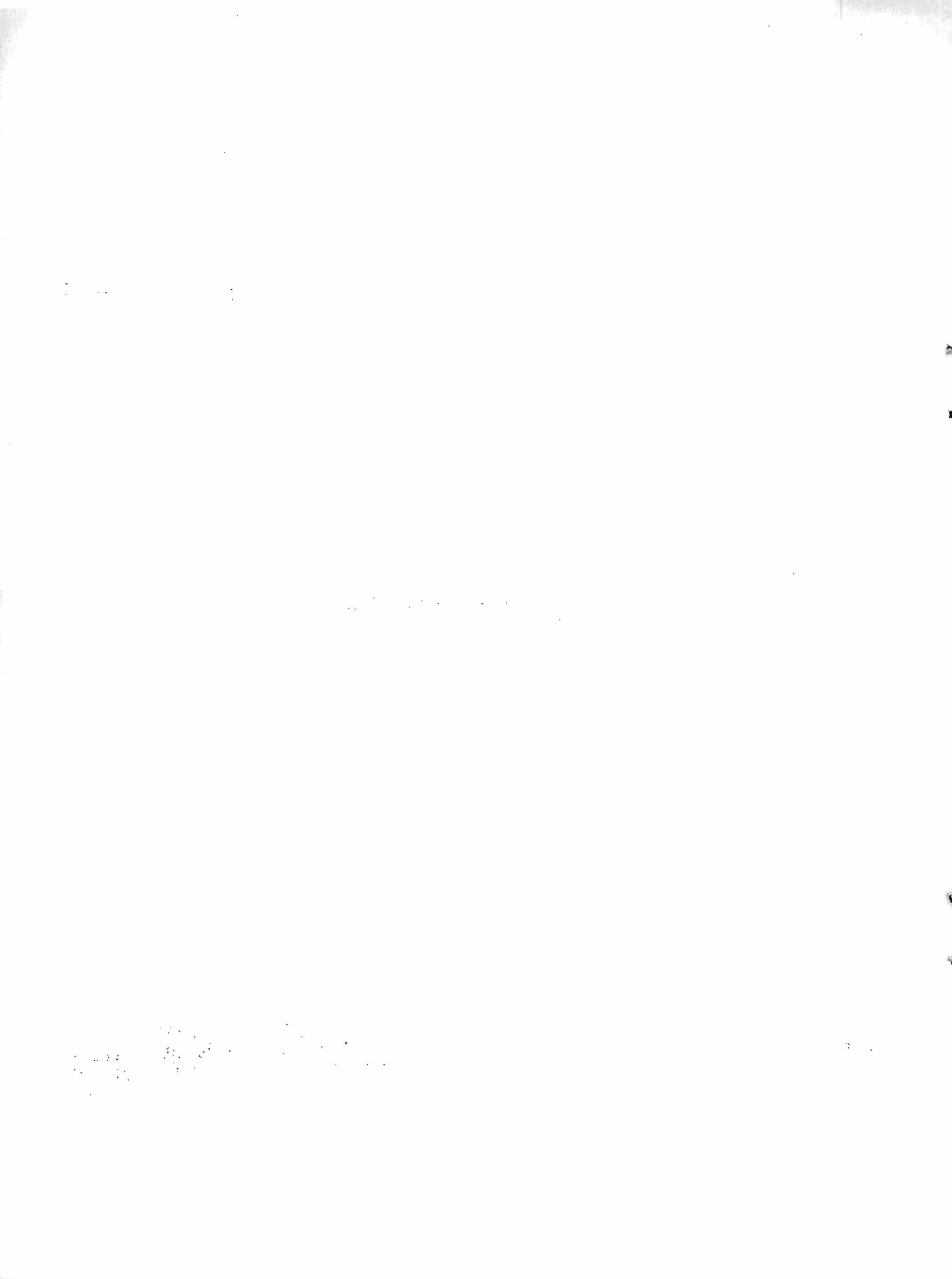
Documentos de Seminario



LENGUAJES Y PROGRAMACION

DS/1
100
1975

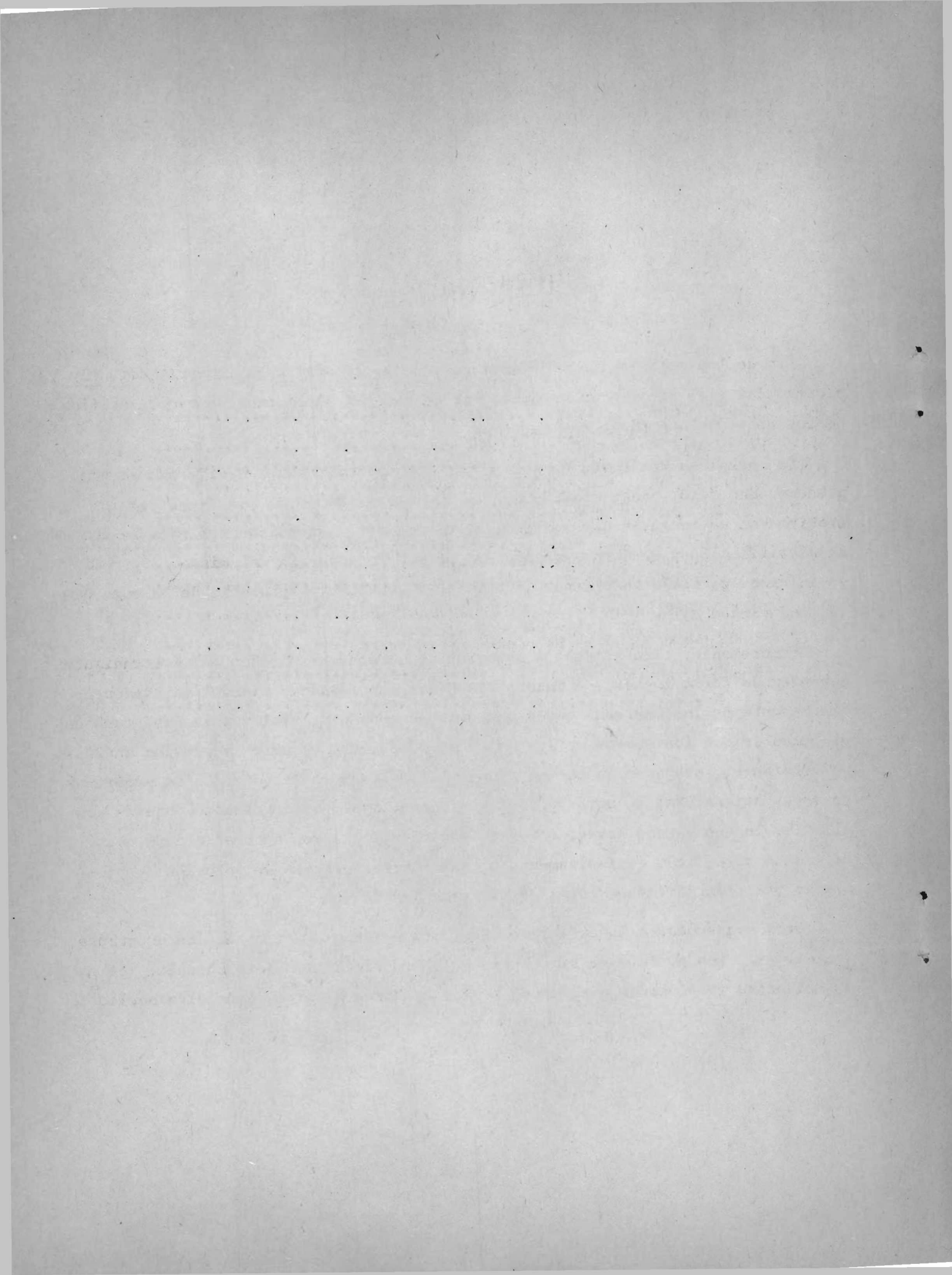
CURSO LATINOAMERICANO DE
PROCESAMIENTO DE DATOS (PED)
APLICADO A LAS CIENCIAS SOCIALES





INDICE

	<u>Página</u>
I. INTRODUCCION	1
II. ALGORITMOS Y DIAGRAMAS DE FLUJO	2
1. Definición de algoritmos	2
2. Diagramas de flujo y aplicaciones	3
III. ¿QUE ES UN PROGRAMA?	34
IV. LENGUAJES	35
1. Lenguaje de máquina	35
2. Lenguaje simbólico	38
V. ESTRUCTURACION DE PROGRAMAS Y MODULARIDAD	52
VI. PROBLEMAS RESUELTOS	56
VII. PROBLEMAS PROPUESTOS	64
BIBLIOGRAFIA	66



I. INTRODUCCION

Uno de los principales problemas con que se encuentra el investigador, el programador o el usuario de computadores en general es la representación gráfica de los procesos que tiene que resolver.

Lo normal es que trate de solucionar las dificultades a medida que se van presentando, pero, desgraciadamente, lo hace en la etapa de escritura del problema en un lenguaje que entienda el computador, saltándose así toda la etapa de clarificación y de formalización del método de solución del mismo. Es como si iniciara un viaje conociendo la meta pero totalmente ignorante de la ruta que le conducirá a ella.

Seguramente, esto se debe a la falta de cursos que enseñen a los estudiantes a pensar en forma lógica, de manera ordenada, con método y disciplina, independientemente de las carreras que sigan o vayan a seguir. Tal vez se debe también al hecho de que los cursos de lenguajes de computador se hacen a presión en las universidades, tratando de que el alumno los aplique en la solución de problemas de otras asignaturas lo antes posible, sin que haya una coordinación previa con ellas y, lo que es más grave, sin que se emplee el tiempo necesario para que el alumno se dedique exclusivamente a construir algoritmos de solución de problemas y a plantearlos en forma de diagramas de flujo.

Otra consecuencia lamentable es la falta de documentación de los programas y sistemas. Los puntos que siguen tienen como principal objeto corregir las deficiencias ya anotadas o al menos servir de ayuda a quien desee eliminarlas.

II. ALGORITMOS Y DIAGRAMAS DE FLUJO

1. Definición de algoritmos

Se define el algoritmo como un conjunto finito de pasos que permiten obtener la solución de un problema. Existen algoritmos numéricos y no numéricos. Ejemplos del primer tipo son: los de las operaciones aritméticas, el algoritmo para obtener la raíz cuadrada de un número, el algoritmo para resolver un sistema de ecuaciones, etc. Ejemplos del segundo tipo son: las recetas de cocina, las instrucciones necesarias para cambiar un neumático a un auto, el algoritmo que permite colorear un mapa con sólo cuatro colores, etc.

Para aclarar mejor el concepto de algoritmo, se tienen los dos ejemplos siguientes:

Ejemplo 1. Se tienen escritos en una hoja, cuatro números distintos, enteros, desordenados. Determinar el mayor de ellos.

Si se observan los números escritos, se puede señalar, casi de inmediato, el número mayor. Pero este resultado se ha obtenido al realizar un proceso mental, del cual algunos pasos se efectúan casi inconscientemente. Al analizar detenidamente el proceso efectuado y al escribir los pasos dados, se obtiene el siguiente algoritmo:

- i) Se compara el primer número con el segundo
- ii) El que resulte mayor se compara con el tercero
- iii) El que resulte mayor de esta segunda comparación se compara con el cuarto
- iv) El que resulte mayor de esta tercera comparación es el número pedido.

Ejemplo 2. Dar las instrucciones a una persona para que sirva una taza de café con leche. Dicha persona no sabe hacerlo pero cuenta con todos los elementos necesarios para ello, esto es, loza y servicio limpios, café y leche calientes, azúcar.

El algoritmo respectivo será:

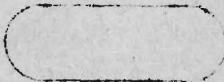
- i) echar azúcar en la taza
- ii) echar café a continuación
- iii) echar leche.

A pesar de que este algoritmo permite preparar una taza de café con leche, con él se pueden obtener múltiples resultados que dependerán del criterio y gusto de la persona que sirva. Luego si se quiere una solución determinada, se deben especificar con más detalle los pasos del algoritmo. Podría ser por ejemplo:

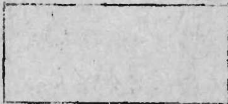
- i) echar tres terrones de azúcar en la taza
- ii) echar café hasta un cuarto de taza
- iii) echar leche hasta llenar la taza.

2. Diagramas de flujo y aplicaciones

El diagrama de flujo es la representación gráfica del algoritmo de solución de un problema. Para construirlo se cuenta con determinadas figuras geométricas o "símbolos". Así, por ejemplo, la figura siguiente:

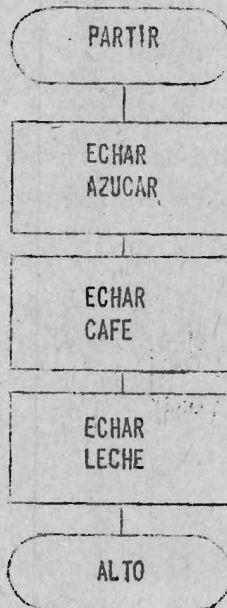



que permite iniciar o terminar un proceso, según la leyenda que se coloque en su interior.

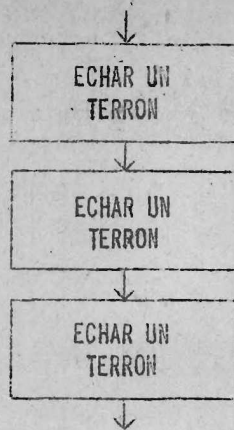


El rectángulo que permite indicar la operación que se va a realizar.

Con estos dos símbolos se puede hacer el diagrama de flujo correspondiente al algoritmo del ejemplo 2. Al considerar el algoritmo más general se tiene:



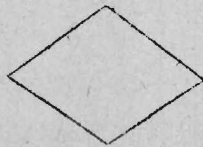
La operación  puede ser reemplazada por:



Pero aún podría ser representada mediante la introducción de un "ciclo".

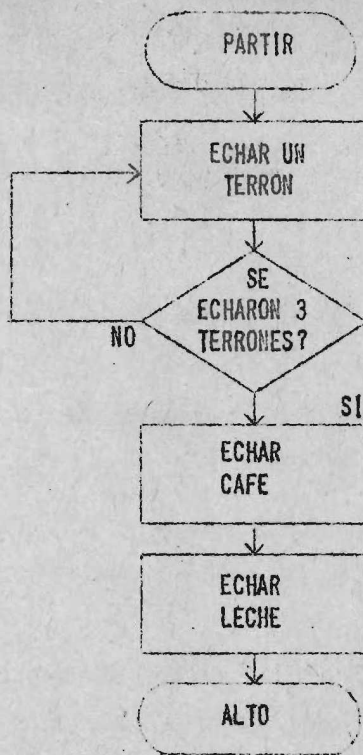


Al salir del bloque, la dirección de la flecha indica un retorno incondicional a la misma operación. Sin embargo, esto se ha transformado en un ciclo indefinido, sin término, pues no existe un elemento que permita detener la operación. Se debe establecer un límite y ese es la cantidad de azúcar que se debe echar. En otras palabras, debe haber un control, en forma de pregunta que indague si se han echado tres terrones. Si la respuesta es NO el ciclo debe continuar, pero si es afirmativa debe terminar, o lo que es lo mismo, salir de él para realizar la operación que sigue en secuencia. Esto se interpreta también como una "decisión" tomada a partir de las preguntas y sus posibles respuestas y en el diagrama de flujo se representa con un rombo:



que permite encerrar en él una pregunta

Utilizando este nuevo símbolo, se obtiene el diagrama de flujo que se indica a continuación:

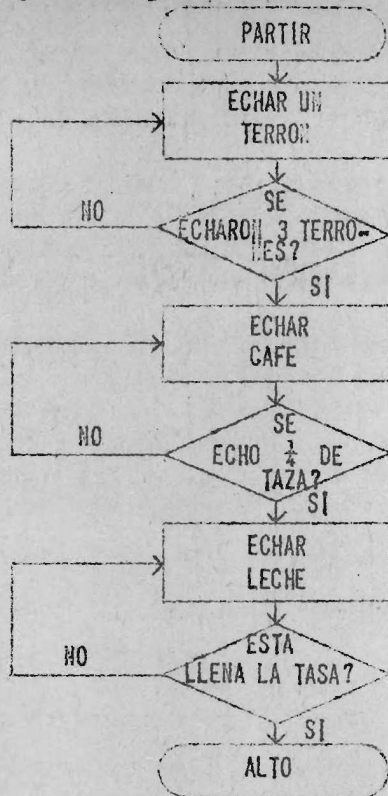


Observando el ciclo, se verifica que se ejecuta mientras no se hayan echado 3 terrones; una vez que esto ocurre, se sale de él para pasar a la operación "ECHAR CAFE".

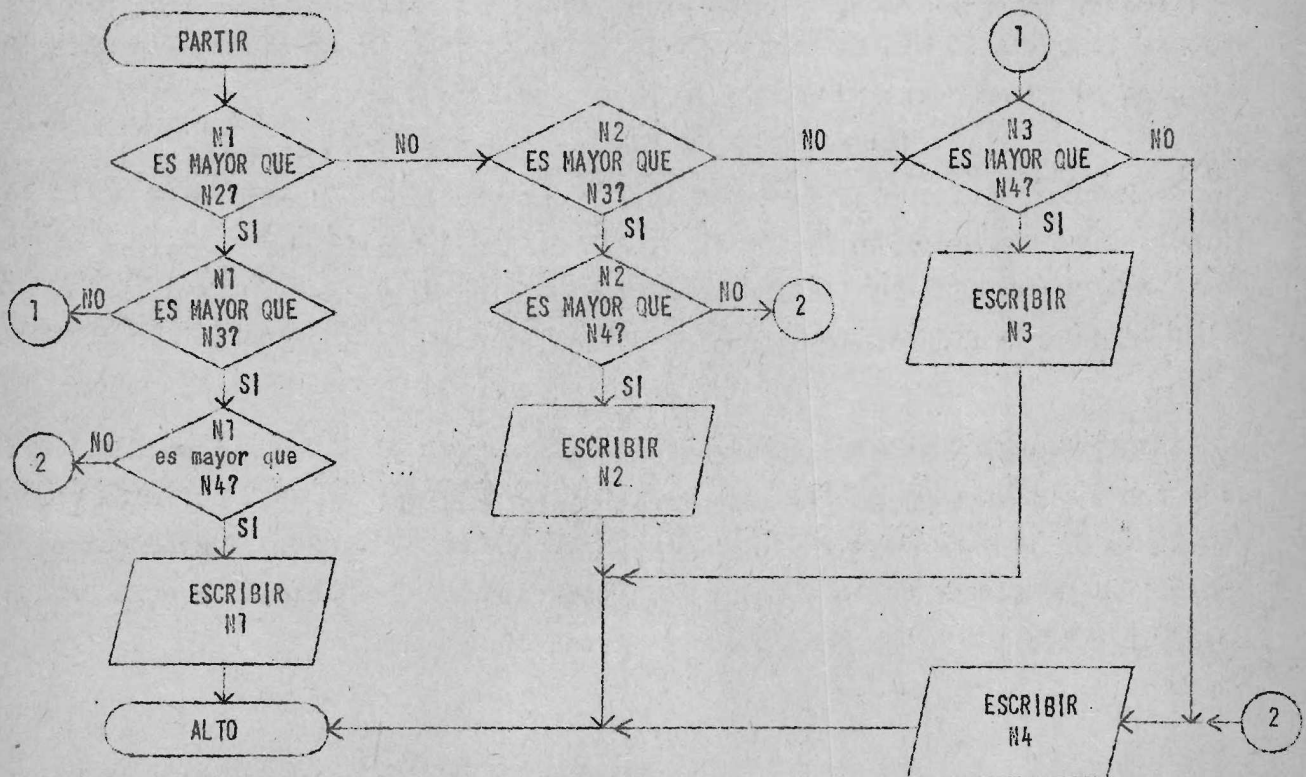
De estos diagramas se pueden deducir algunas reglas sobre las cuales se insistirá para evitar ambigüedades en la construcción de otros. Debe advertirse que de una figura de iniciación (PARTIR) sólo puede salir una flecha; a una figura de término (ALTO) pueden llegar múltiples flechas y desde luego, no puede salir ninguna. A las figuras de operación pueden llegar una o más flechas pero puede salir sólo una.

Finalmente, a la figura de decisión pueden llegar una o más flechas y salen tantas como sea el número de respuestas posibles. Evidentemente, este número debe ser superior a uno para que exista decisión.

Volviendo al problema, el diagrama de flujo completo será:



Y el diagrama de flujo correspondiente al ejemplo 1 será el siguiente:



Se han introducido dos nuevas figuras en el diagrama de flujo:



El paralelogramo, que se utiliza para indicar lectura o escritura de información, o lo que es lo mismo, ENTRADA o SALIDA de información en el proceso, y



El círculo, que permite enlazar dos o más partes del diagrama, dentro de una misma página. En el interior del conector se especifica cualquier símbolo, el que debe repetirse en el o los conectores de enlace.



Existe también el conector, que permite enlazar dos o más partes del diagrama, que están en distintas páginas. En este caso, dentro del conector se especifican dos símbolos separados entre sí por: - ó / ó , ó ; ó cualquier otro carácter especial. El primer símbolo identifica la página y el segundo al conector dentro de la página.

Se puede observar en el diagrama que las flechas no necesariamente tienen que "entrar" en una figura, pueden empalmar con otra flecha. Lo importante es que el flujo quede claro.

Una de las unidades que componen un computador es la MEMORIA, formada por "celdas" donde se guarda información. Esas celdas están numeradas en forma correlativa y el número de orden constituye la "dirección" de la celda. Se considerará el problema siguiente: Guardar en la celda 10 el resultado de la suma del contenido de la celda 15 más el contenido de la celda 20. Si se utiliza la notación () para expresar "contenido de", se puede escribir:

$$(celda\ 10) = (celda\ 15) + (celda\ 20),$$

que se puede leer: el contenido de la celda 10 es igual al contenido de la celda 15 más el contenido de la celda 20. Pero, la interpretación del signo = es un tanto ambigua puesto que se podría interpretar como: lo que "había" en la celda 10 es igual a lo que se puede obtener si se suman los contenidos de las celdas 15 y 20.

Ese problema queda solucionado si se interpreta el signo = como "está definido por", con lo cual se lee: el contenido de la celda 10 está definido por el resultado de la suma de los contenidos de las celdas 15 y 20. También se puede decir: el resultado de la suma de los contenidos de las celdas 15 y 20 "se asigna a" la celda 10, o "se guarda en" la celda 10.

Otros símbolos que se utilizan corrientemente para definir o asignar son: la flecha en sentido de derecha a izquierda (\leftarrow) y dos puntos seguidos del signo igual ($:=$). En ese caso se anotará:

$$\begin{array}{l} \text{(Celda 10)} \leftarrow \text{(celda 15)} + \text{(celda 20)} \text{ ó} \\ \text{(Celda 10)} := \text{(celda 15)} + \text{(celda 20)} \end{array}$$

El símbolo de definición permite también escribir expresiones como la siguiente:

$$\text{(celda 10)} = \text{(celda 10)} - \text{(celda 15)}$$

que significa: el contenido de la celda 10 queda definido por el resultado obtenido al restar del contenido de la celda 10, el contenido de la celda 15. Si dichos contenidos fueran 1345 y 826, el nuevo valor guardado en la celda 10 será 519.

Si se generaliza y se identifican con los nombres A, B y C las celdas 10, 15 y 20 respectivamente, se podrá escribir:

$$A = B + C,$$

que se puede interpretar como: la variable A queda definida con el resultado obtenido al sumar el valor de la variable B al valor de la variable C.

Se llama la atención al hecho de haber asignado las variables A, B y C a tres celdas de memoria.

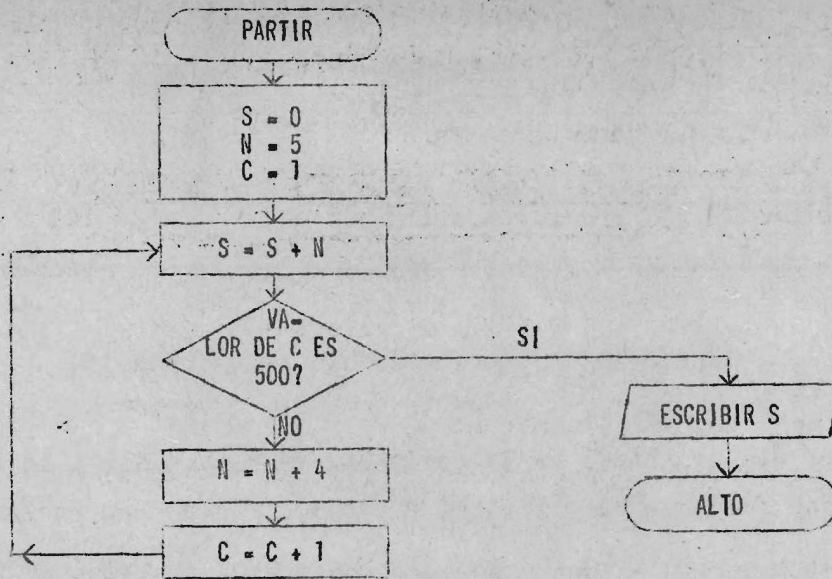
El siguiente problema se resolverá con estos nuevos conceptos:

Ejemplo 3. Hallar la suma de 500 términos de la progresión aritmética cuyo primer término es 5 con incremento 4, esto es 5, 9, 13, 17, ... Se usarán las variables:

S para acumular la suma

N para formar cada término

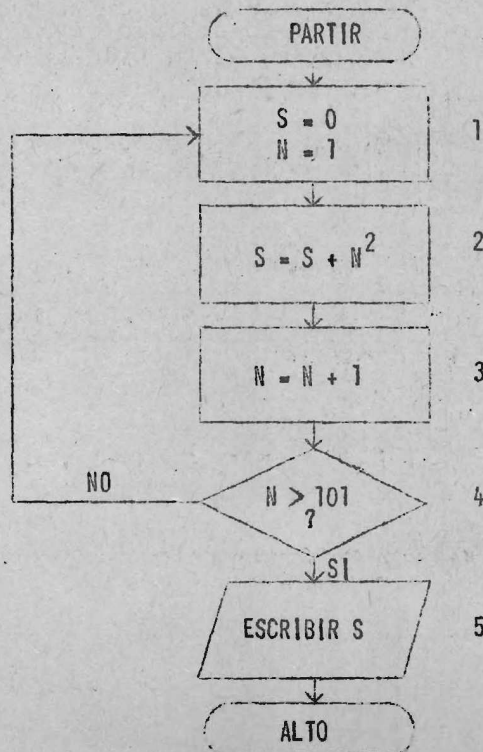
C para controlar el número de veces que se realiza el ciclo de suma.



Ejemplo 4. Se desea tener un algoritmo que permita encontrar la suma de los cuadrados de los primeros 101 enteros positivos. En otras palabras, se quiere encontrar el valor de: $S = 1^2 + 2^2 + 3^2 + \dots + 101^2$.

Con N se designará sucesivamente a los números 1, 2, 3, ..., 101. Con cada valor de N se calculará su cuadrado y este valor incrementará la suma acumulativa S. Por supuesto, S debe tener un valor inicial 0.

En el ejemplo anterior se utilizó C para controlar la repetición del ciclo. Ahora se aprovechará la misma variable N y el valor que ella toma para efectuar el control.



Si se desea verificar si el algoritmo funciona, se numeran los bloques, se considera un número menor de términos, por ejemplo 5, y se construye la tabla siguiente:

Paso	Bloque Nº	Valor de las variables		Control	Sí o No
		S	N		
Partir					
1	1	0	1		
2	2	1			
3	3		2		
4	4			2 > 5?	No
5	2	5			
6	3		3		
7	4			3 > 5?	No
8	2	14			
9	3		4		
10	4			4 > 5?	No
11	2	30			
12	3		5		
13	4			5 > 5?	No
14	2	55			
15	3		6		
16	4			6 > 5?	Sí
17	5	Escribir S			
Alto					

Nótese que éstos diagramas de flujo no sugieren la idea de resolver el problema con un computador, a pesar de que se han utilizado las características de su "memoria" para avanzar en los conceptos que permiten mejorar la construcción de ellos. Esto significa que el problema puede resolverse perfectamente con cualquier sistema de procesamiento de datos, incluyendo lápiz y papel, como se acaba de realizar para controlar el funcionamiento del algoritmo de solución del problema propuesto.

Con los elementos hasta ahora estudiados se puede examinar un problema frecuente, cual es el de calcular medidas estadísticas de un juego de datos.

Ejemplo 5. Calcular la media aritmética.

El algoritmo de solución será:

- i) Poner en cero los acumuladores para la suma y el contador de lecturas.
- ii) Leer los datos.
- iii) Contabilizar la lectura y sumar el dato al acumulador.
- iv) Volver al paso ii).

Se ha llegado a un punto en que se entra a un ciclo que parece infinito, dado que siempre se retorna a leer datos. A pesar de que no se visualiza cómo "romper" esa secuencia obligada de operaciones, en la práctica, en algún momento, al tratar de realizar la lectura, se encontrará que los datos se han terminado y este hecho permitirá salir del ciclo.

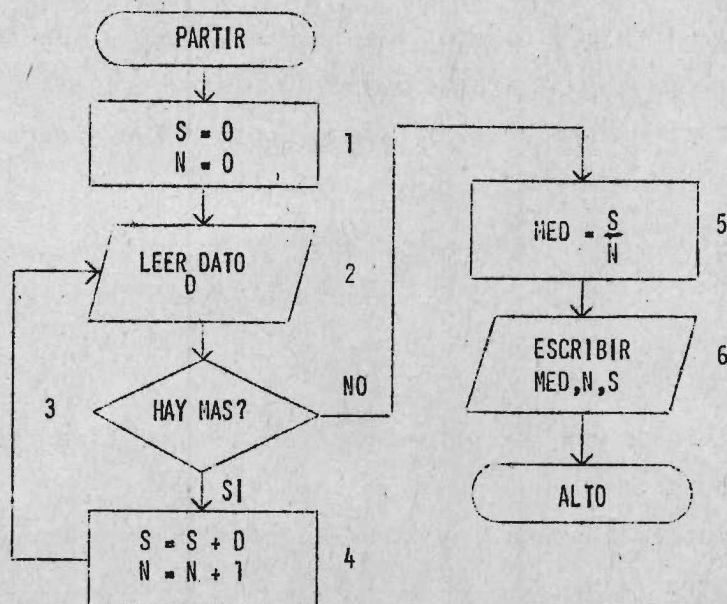
Cualquiera que sea el medio en que esos datos estén registrados, siempre es posible hacer la pregunta ¿hay más datos?, obviamente, esta consulta debe ser hecha después de la lectura, para saber si al efectuar esta operación se encontraron datos o no.

Es lo que hace el ser humano al leer datos en una hoja de papel. La vista recorre el conjunto transmitiendo la información al cerebro. Por cada dato leído, el cerebro consulta ¿hay más? y al recibir respuesta afirmativa ordena leer nuevamente. ¿Cuándo sabrá que no hay más? Cuando la vista se haya dirigido al espacio a continuación del último y haya transmitido la información encontrada. Esa información, que puede ser: espacio en blanco, un paréntesis, un punto, etc., le indicará al cerebro que no hay más datos.

En el problema planteado los pasos que están a continuación del ciclo son:

- v) Calcular la media aritmética.
- vi) Escribir la media, el número de datos y la suma acumulada.

El diagrama de flujo correspondiente a este algoritmo es:

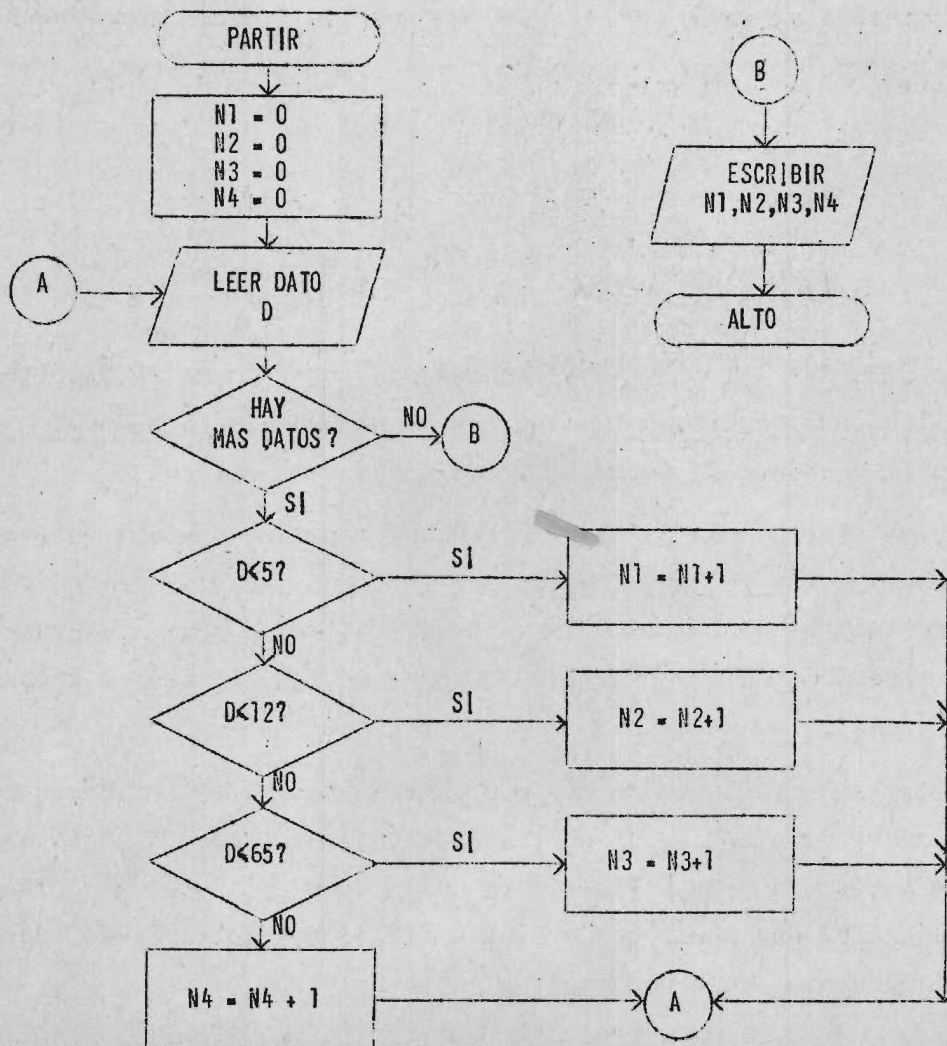


Es conveniente hacer notar que, cuando se especifican dos o más operaciones dentro de un bloque, como es el caso de los bloques 1 y 4 del diagrama anterior, la secuencia de ejecución de ellas se efectúa en el mismo orden en que están escritas. Esta secuencia, que siempre se mantiene, es independiente del punto de entrada de las flechas del bloque.

Ejemplo 6. Considerando el mismo juego de datos del ejemplo anterior, calcular la frecuencia de ellos en cada una de las 4 siguientes categorías, según su valor numérico.

- menores que 5
- de 5 a 11
- de 12 a 64
- mayores de 64

Para la solución, se han empleado los símbolos N1, N2, N3 y N4 para calcular las frecuencias de la primera, segunda, tercera y cuarta estratificación, respectivamente.



En aquellos problemas en que se trabaja con conjuntos de datos, se les asignan nombres a dichos conjuntos con el objeto de identificarlos. El nombre asignado puede ser totalmente arbitrario o puede indicar o dar idea referente al tipo de datos que contiene el conjunto. Por ejemplo, si se tienen las estaturas de niños de 7 años, se las podrá identificar con: X, ESTAT, ESTATURA7, etc. Todos ellos son nombres posibles, sin embargo, ESTAT es más claro que X y ESTATURA7 es mucho más que los anteriores, en relación con el tipo de información contenida en el conjunto.

Si se desea identificar cada dato, será necesario indicar primero el nombre del conjunto y en seguida la posición o número de orden del dato dentro de él. Por ejemplo, sea el conjunto de temperaturas máximas que hubo en una ciudad en un lapso de 5 días:

34,3 32,1 29,7 31,5 32,4

Si se identifica al grupo con el nombre genérico T, cada temperatura tendrá el nombre particular T_i en que i varía de 1 a 5. Así se obtiene:

$$T_1 = 34,3$$

$$T_2 = 32,1$$

$$T_3 = 29,7$$

$$T_4 = 31,5$$

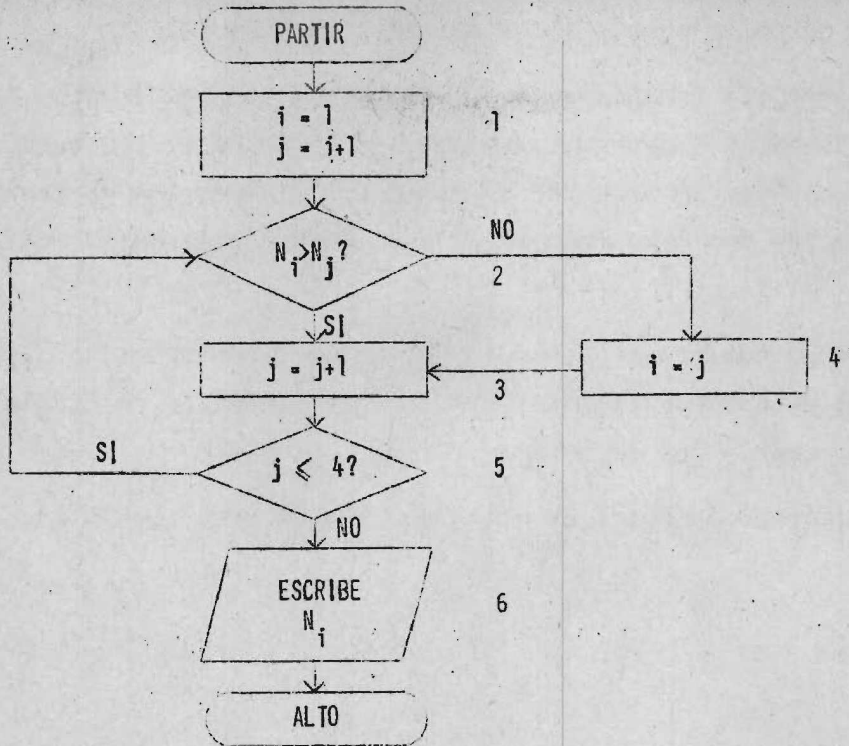
$$T_5 = 32,4$$

El número de orden del dato se llama índice, los nombres particulares, variables con índice o variables indexadas y el nombre genérico, nombre de arreglo siendo arreglo el conjunto de datos.

El índice se escribe más bajo que el nombre genérico, de ahí que corrientemente se le denomina sub-índice. Cuando se tiene que escribir el índice a la misma altura del nombre genérico es conveniente diferenciarlo en alguna forma de él. Puede ser escribiéndolo con minúscula, o de un tamaño menor o separándolo con algún signo especial.

La razón de esta medida queda clara al considerar el elemento general de un conjunto o, como se dijo antes, un nombre particular tal como T_i . Si el índice se escribe con mayúscula a la misma altura de T, queda TI, que se confunde con el nombre de variable TI que puede haber sido utilizado, o lo será más adelante, en otra parte del diagrama, como nombre simple de variable.

Al resolver el ejemplo 1 haciendo uso de índices, se tendrá:



Para controlar el funcionamiento del algoritmo se asignan valores arbitrarios a las variables N_i . Sean ellos 5, 1, 9 y 13 para definir N_1 , N_2 , N_3 y N_4 respectivamente.

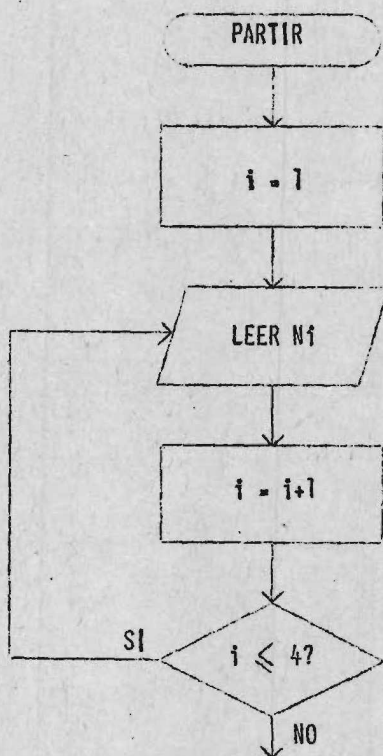
Peso	Bloque Nº	Valor de las variables				Control	Sí o No
		i	j	N_i	N_j		
Partir							
1	1	1	2	5	1		
2	2					$5 > 1$	Sí
3	3		3		9		
4	5					$3 \leq 4$	Sí
5	2					$5 > 9$	No
6	4	3		9			
7	3		4		13		
8	5					$4 \leq 4$	Sí
9	2					$9 > 13$	No
10	4	4		13			
11	3		5		?		
12	5					$5 \leq 4$	No
13	6	Escribe		N_4			
Alto							

En el paso 11, aun cuando el índice j está "apuntando" a N_5 , que no existe, no tiene importancia porque dicha variable no se utiliza.

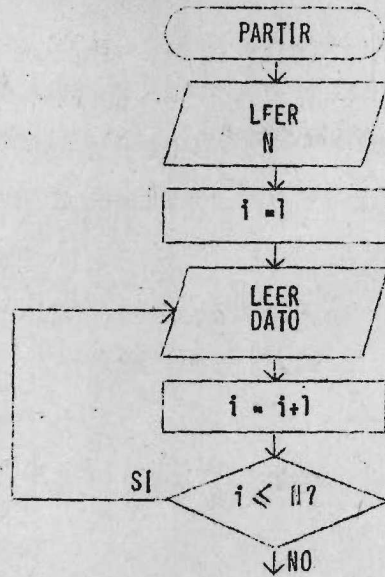
Se han seguido construyendo los algoritmos independientes de la idea de resolver los problemas mediante computador. En lo sucesivo las soluciones se orientarán por ese camino. A pesar de ello, se podrá notar que el problema podrá ser siempre resuelto con otro sistema de procesamiento de datos utilizando el mismo algoritmo.

De acuerdo con lo anterior, se presentan a continuación algunas variantes del diagrama visto para el ejemplo 1. Estas variantes suponen almacenamiento de datos en la memoria del computador.

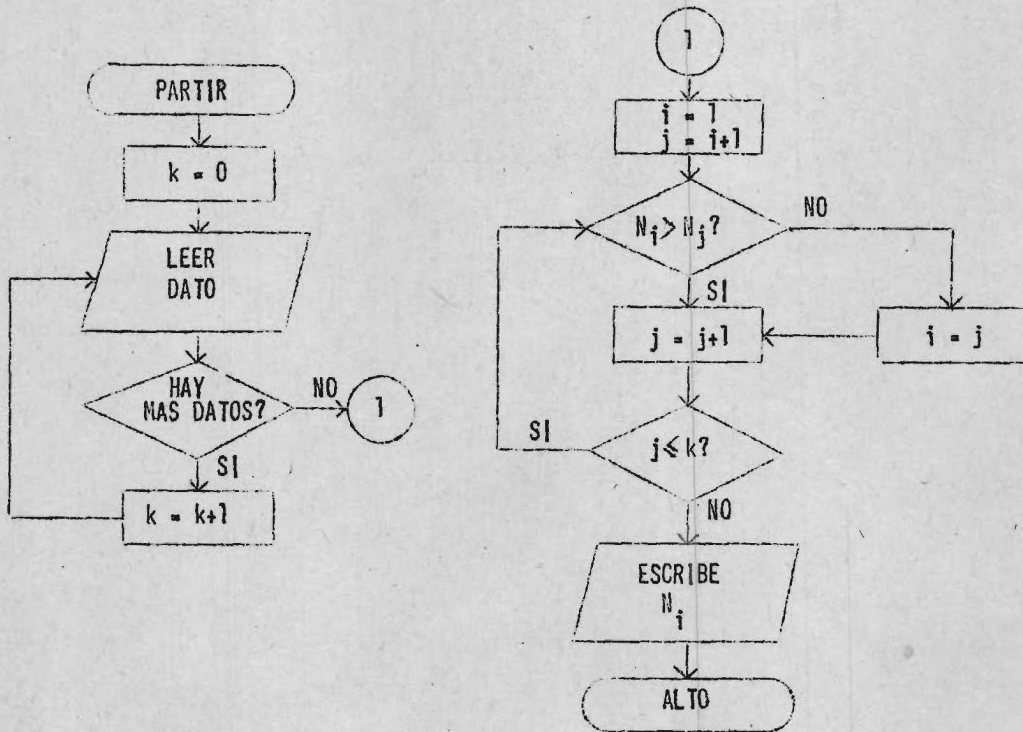
a) El número de datos es conocido (en el problema es 4).



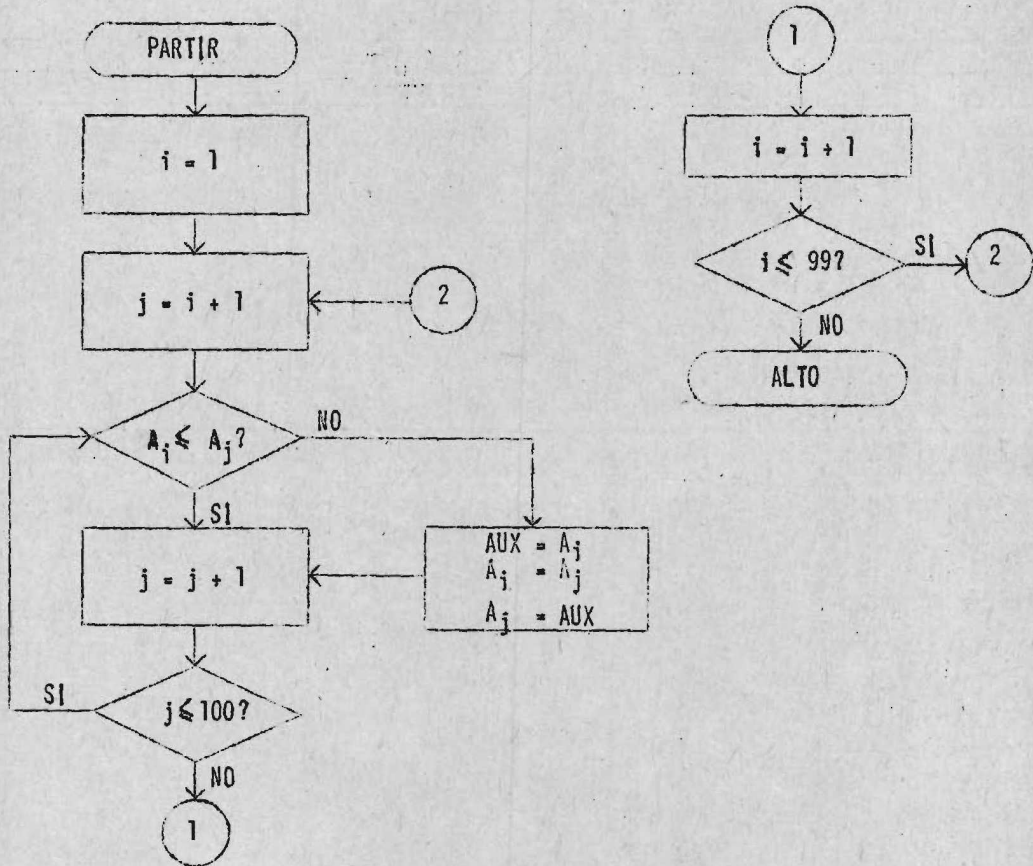
b) El número de datos no es conocido, pero se sabe que encabeza el conjunto de datos (N será el número de datos).



c) El número de datos no es conocido.



La solución b) es más general que la a), y la c) más que la b). Esto no significa que sea la única ni tampoco que es la mejor posible. De idéntica manera a la usada para almacenar información en memoria, se pueden sacar datos de ella. Ejemplo 7. Se tiene un arreglo de 100 elementos, que se desean ordenar de menor a mayor. Con el objeto de simplificar los algoritmos, se omitirán los ciclos de lectura e impresión.



Se han resuelto dos problemas mediante el uso de variables con un índice. Sin embargo, este tipo de variables es ineficiente para resolver el problema del ejemplo siguiente:

Ejemplo 8. Se tienen registrados los datos de una población y entre ellos figuran la edad y el estado civil de cada habitante. Se pide obtener la siguiente tabla:

Edad	Población por estado civil según edades simples						
	Estado Civil						
	Soltero	Casado	Conviviente	Separado	Divorciado	Viudo	Ignorado
1 año y menos							
37 años							
98 años							
99 años y más							
Ignorado							

Para resolver el problema debe considerarse que se ha hecho la siguiente codificación:

Edad ignorada	100
Estado civil	
Soltero	1
Casado	2
Conviviente	3
Separado	4
Divorciado	5
Viudo	6
Ignorado	7

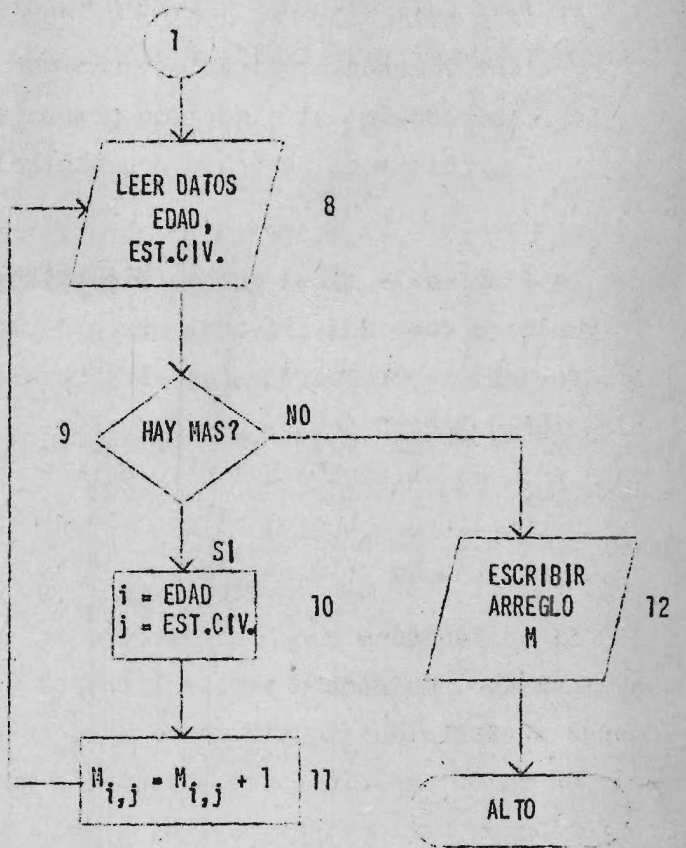
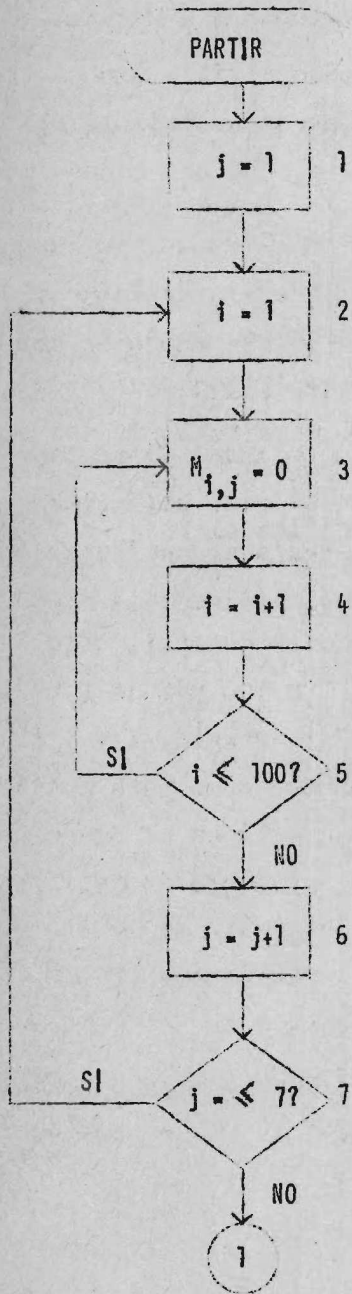
Se puede observar que la tabla contendrá 700 resultados correspondientes a 100 edades por 7 estados civiles, por lo tanto será necesario utilizar variables con índice para acumular cada uno de ellos. En caso contrario, se tendrían que crear 700 variables y darle un nombre distinto a cada una, lo que no permitiría un algoritmo eficiente, dado que sería necesario construir un ciclo de acumulación por cada variable.

Es posible resolver el problema haciendo uso del tipo de variable "indexada" que se ha visto, o sea variable con un índice. Pero la variable con un índice implica un arreglo lineal, sea renglón o columna, y la tabla pedida corresponde a un arreglo bilineal, es decir, de dos dimensiones. Se tendrá que trabajar entonces con 7 arreglos lineales, uno por cada tipo de estado civil. Sin embargo, la construcción del algoritmo presenta aún dificultades que es posible eliminar si el problema se resuelve con variables con dos índices, uno por cada dimensión del arreglo.

La idea es la misma que se vió para variables con un índice. Habrá un nombre genérico que identifica al arreglo y nombres particulares que identifican al dato. El nombre particular está formado por el nombre genérico seguido de la ubicación del dato dentro del arreglo. Cuando se trata de un arreglo de dos dimensiones, la ubicación del dato corresponde al cruce del renglón y de la columna que lo contienen. Luego, si con el primer índice se identifica el renglón y con el segundo la columna, se tendrá la posición del dato.

Si se denomina con M al arreglo del problema, el elemento general, o lo que es lo mismo, un nombre particular sera M_{ij} , i variará de 1 a 100 y j de 1 a 7. Cuando se trate de identificar a un determinado elemento del arreglo, i y j tendrán un valor numérico, será necesario entonces separarlos por coma para evitar confusión en su lectura. Ej. $M_{5,4}$ es el nombre del dato que está en el cruce del renglón 5 con la columna 4, diferente de M_{54} que es el elemento 54 del arreglo lineal M .

Solución del problema:



Observaciones:

a) Si se hubiera colocado el ciclo que se repite más veces encerrando al ciclo que se repite menos, el número de bloques que se obtiene será el mismo. Sin embargo, la cantidad de veces que se ejecuten las operaciones indicadas en los bloques 2,6 y 7 será bastante mayor.

b) Los bloques 10 y 11 pueden ser reemplazados por uno solo que contenga $M_{EDAD}, ESTCIV = M_{EDAD}, ESTCIV + 1$

c) En esta solución no se ha almacenado el conjunto de datos en memoria, de tal manera que en el bloque 8 no ha sido necesario utilizar variables con índice. En todo caso, a diferencia de los problemas anteriores, ahora se han almacenado dos datos con cada orden de lectura, edad y estado civil.

d) La forma en que debe ordenarse la información y cómo deben darse las instrucciones de escritura para obtener la tabla tal como ha sido solicitada, esto es, con títulos, encabezamientos por columna, etc., es materia de capítulos posteriores, de ahí que en el bloque 12, solamente se haya especificado la orden ESCRIBIR ARREGLO M.

Ejemplo 9. Se desea calcular el valor de la función

$$y = ax^2 + b \quad \text{para } a = 1,5 (10,0) 0,5$$

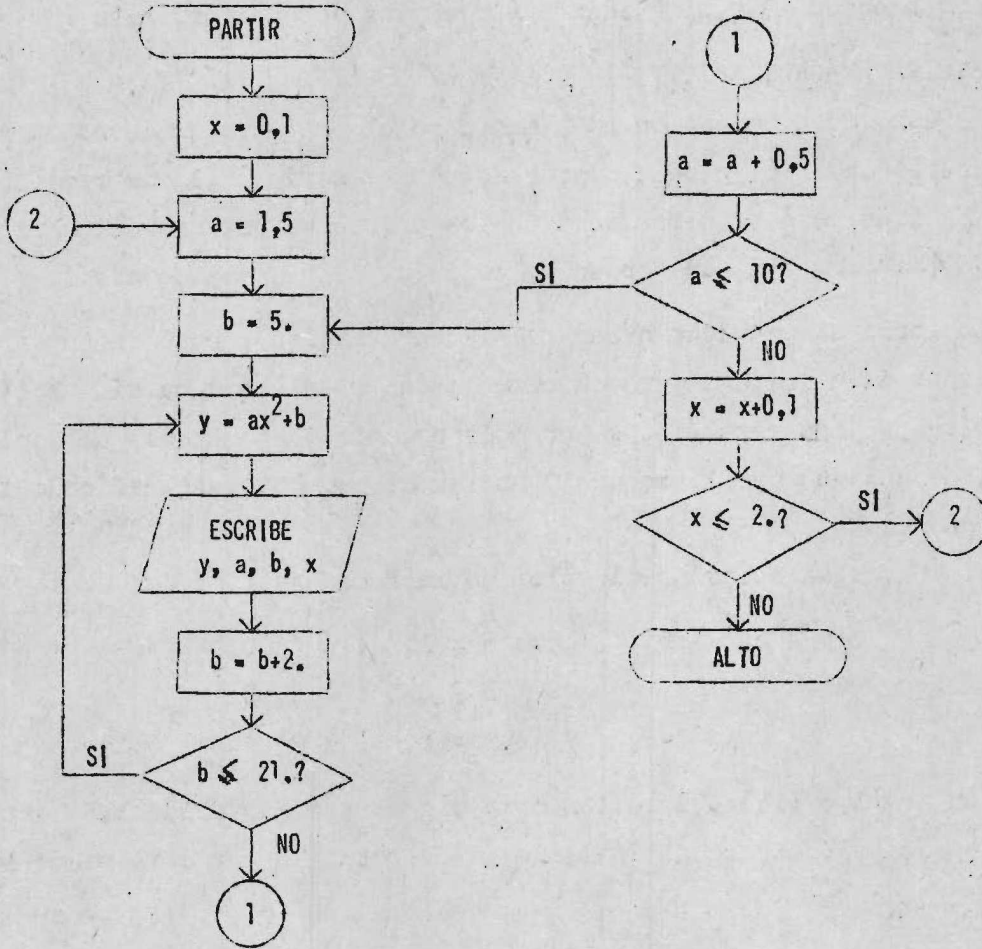
$$b = 0,1 (2,0) 0,1$$

$$x = 5,0 (21,0) 2,0$$

la notación $a = 1,5 (10) 0,5$ se lee como sigue: a varía desde 1,5 hasta 10,0 con incrementos de 0,5. En igual forma se interpretan b y x utilizando los valores que les corresponden.

Lo anterior significa que deben efectuarse todas las combinaciones posibles entre a, b y x. Cada una de esas combinaciones determinará un valor para y.

a) Se calculará el valor de y para una combinación a , b , x , y el resultado se escribirá inmediatamente.



b) Los valores de y , calculados, se guardarán en memoria. Una forma cómoda de resolver el problema es utilizando un arreglo de tres dimensiones. Si se llama Y a dicho arreglo, el elemento general será $Y_{i,j,k}$ en que i controlará la variación de a , j la de b y k la de x .

El límite superior de cada índice se obtiene a base de: valor final, valor inicial e incremento de la variable que controla. Ellos se reemplazan en la fórmula siguiente:

$$\text{lim. sup.} = \text{parte entera de } \frac{(\text{valor final} - \text{valor inicial})}{\text{incremento}} + 1$$

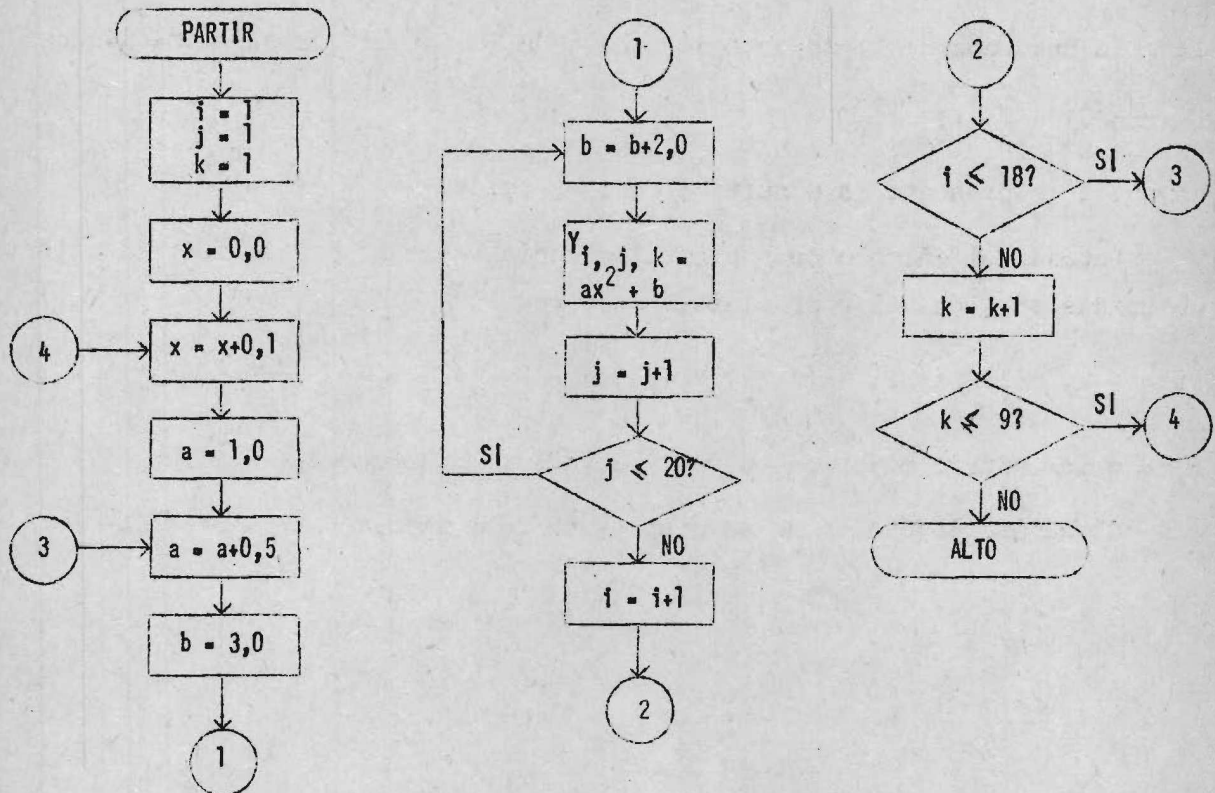
por ejemplo:

$$\text{límite superior de } i = \text{parte entera de } \frac{(100 - 1,5)}{0,5} + 1$$

límite superior de $i = 18$

lo mismo respecto a los índices j y k

Solución del problema:



Los algoritmos se pueden hacer a dos niveles:

- a) a nivel de operaciones, como se ha venido realizando hasta ahora y
- b) a nivel de conjuntos de operaciones.

En el primer caso, se explica en detalle la sucesión de pasos que es necesario dar para obtener el resultado. En el segundo, se sintetizan grupos o "bloques" de operaciones en una sola expresión, de tal manera que se obtiene un algoritmo que indica, en forma general, lo que hay que hacer a través del proceso.

Ejemplo 10. Se tiene un conjunto X de datos. Al comienzo de ellos, se tienen además dos valores m y n, que se utilizarán en la siguiente forma:

$$\text{si } m = 1, \text{ calcular } Y_i = X_i^{\frac{1}{2}}$$

la raíz cuadrada de X_i se obtiene a base de la fórmula de aproximación de

$$\text{Newton } Y = \frac{1}{2} \left(Y_a + \frac{X}{Y_a} \right)$$

en que Y_a representa la penúltima raíz obtenida.

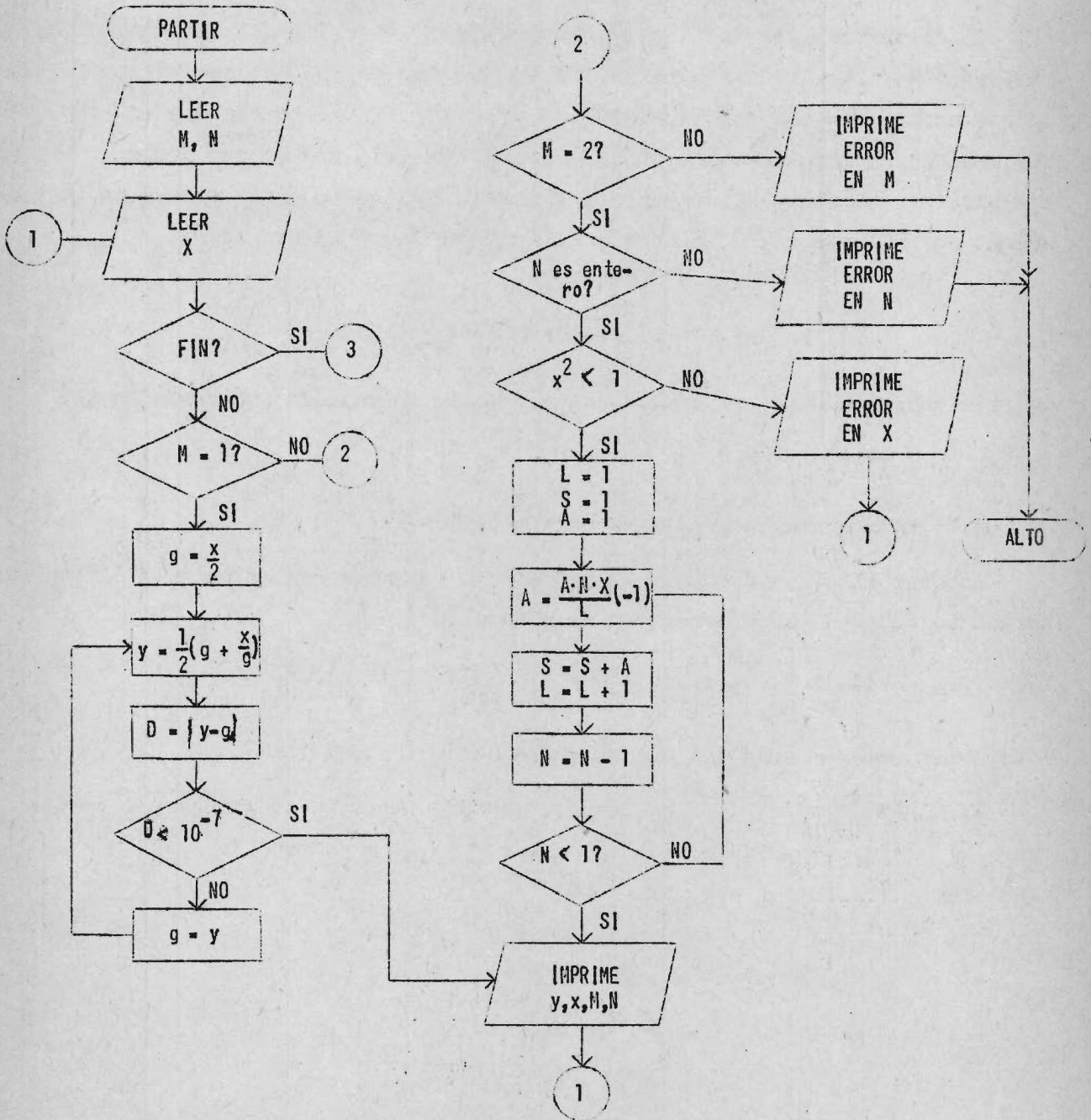
Detener el cálculo cuando la diferencia entre la penúltima y la última raíz obtenidas sea, en valor absoluto, menor que 10^{-7}

$$\text{si } m = 2, \text{ calcular } Y_i = (1 - X_i)^n = 1 - nX_i + \frac{n(n-1)}{2!} X_i^2 - \frac{n(n-1)(n-2)}{3!} X_i^3$$

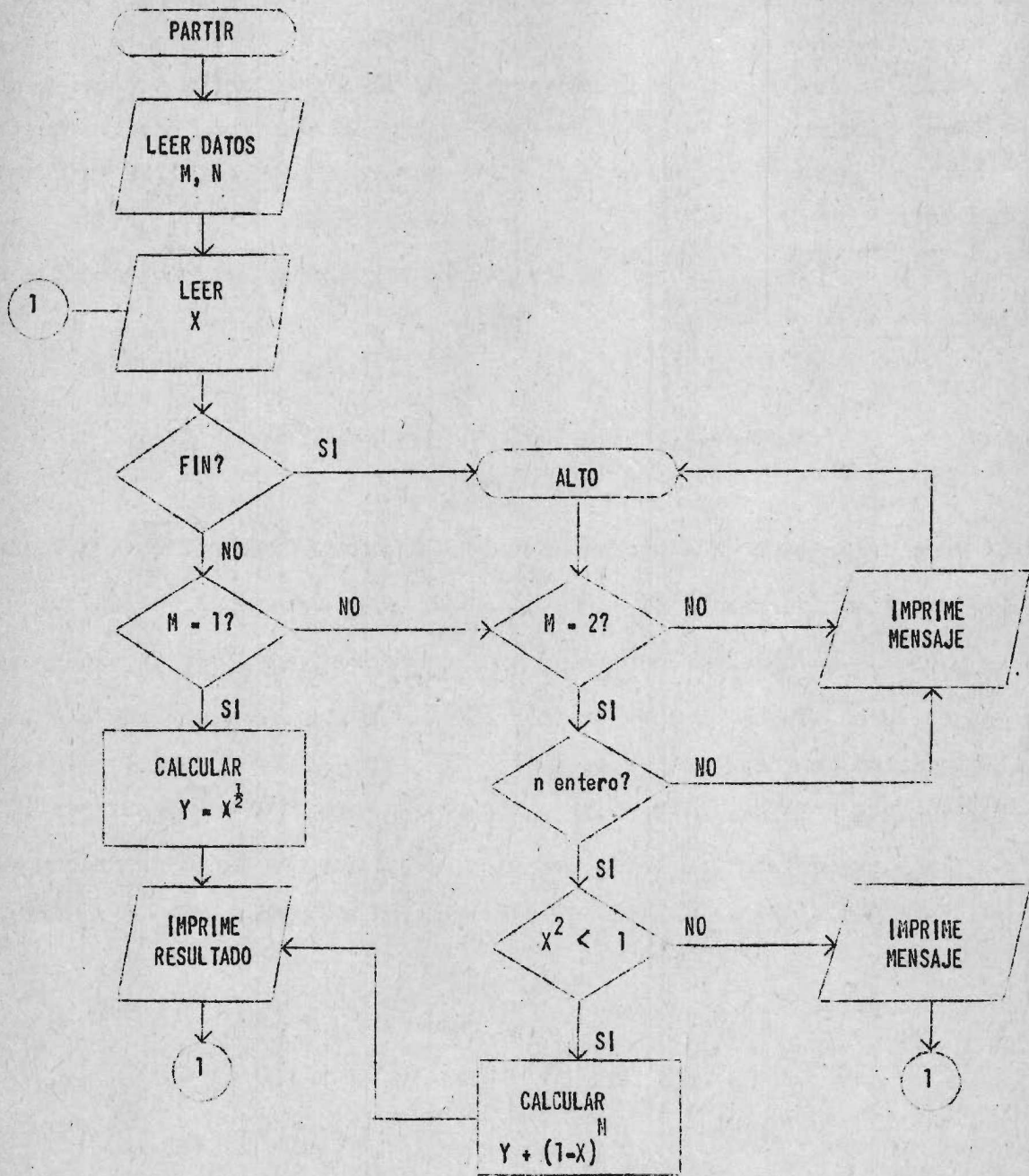
n debe ser entero mayor que 0 y X_i^2 debe ser menor que 1.

Si alguna condición no se cumple, se debe imprimir un mensaje de error.

a) diagrama a nivel de operaciones.



b) diagrama de bloques.



En este diagrama, las expresiones $Y = X^{\frac{1}{2}}$ e $Y = (1-X)^N$ sintetizan bloques de operaciones, lo que permite obtener un diagrama más fácil de leer. Si bien es cierto que no se especifica cómo obtener los valores de Y en cada uno de los casos, se puede controlar mejor la lógica aplicada en el tratamiento de la información. Esto es de bastante utilidad cuando se tienen algoritmos previamente construidos, de manera de intercalarlos solamente en aquellos procesos que los usen; también cuando esas partes del algoritmo se deseen detallar en forma independiente.

Existe la siguiente figura para representar grupos de operaciones que no se detallarán en el diagrama:



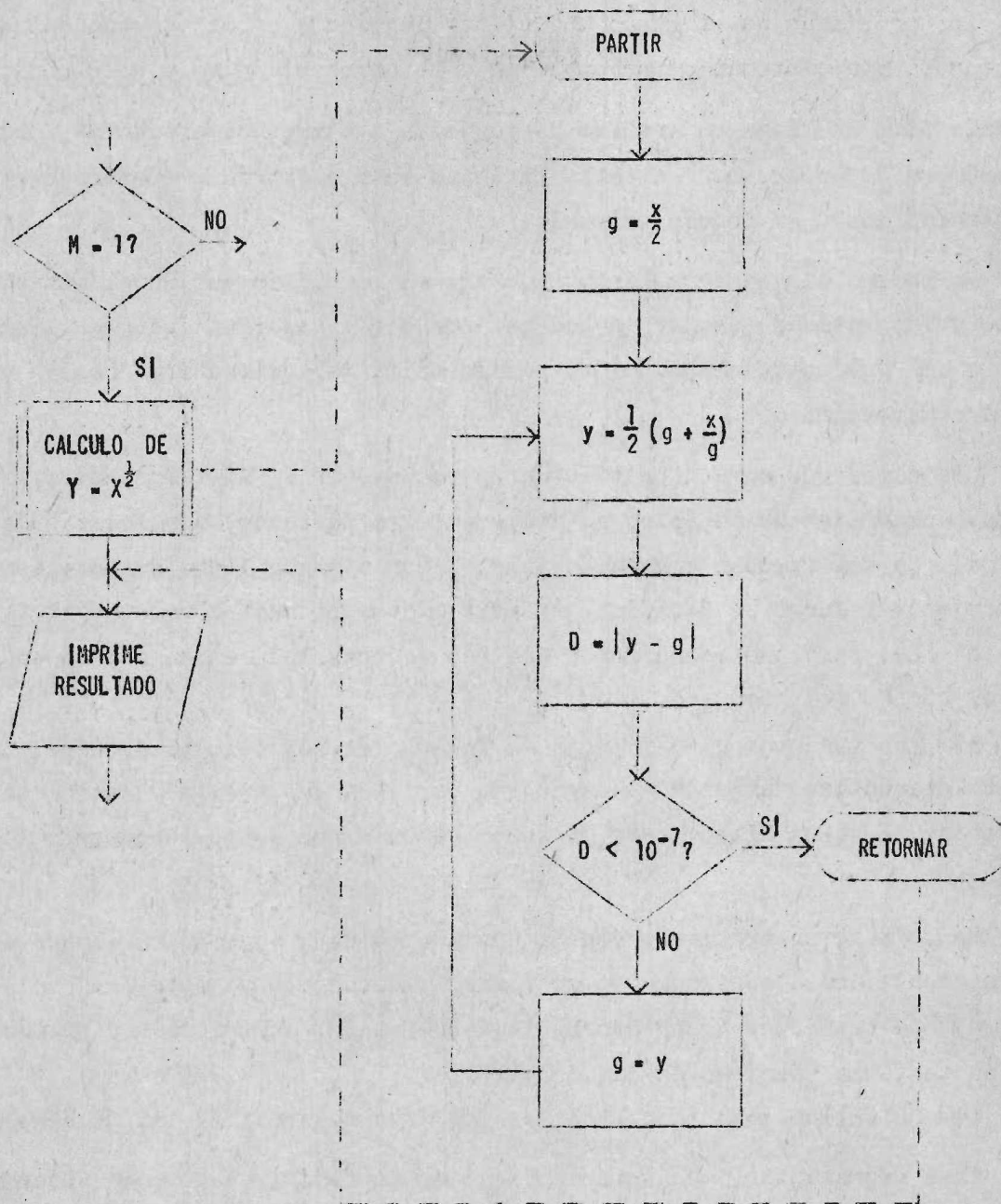
que permite indicar procesos predefinidos.

Si se considera en el diagrama anterior solamente la parte en que figura el cálculo de $Y = X^{\frac{1}{2}}$ se tendrá un ejemplo de uso de un proceso predefinido:

La parte de la derecha en el diagrama corresponde al algoritmo independiente o proceso predefinido que permite obtener $Y = X^{\frac{1}{2}}$. Este algoritmo es "llamado" por el algoritmo principal del cual recibe el dato X . Con X calcula el valor de Y , resultado que entrega al programa principal al retornar a él nuevamente.

La línea segmentada que en la práctica no se usa, se ha colocado con el objeto de hacer más claro el "salto" al proceso predefinido, como, asimismo, el retorno desde él.

c) Representación de un proceso predefinido.



Otro tipo de diagrama que interesa conocer es el diagrama de flujo para sistemas. Con este diagrama se describe el trayecto de la información a través de las distintas unidades que componen el sistema de procesamiento de datos.

Los problemas que figuran a continuación son ejemplos de uso del diagrama mencionado con sistemas de procesamiento de datos mecánico y electrónico.

Ejemplo 11. Se tiene un archivo de tarjetas maestras de productos, ordenadas por número de producto. En estas tarjetas está perforada, además de otros datos, la descripción y el precio unitario.

De bodega llegan formularios que tienen como información el número del producto y la cantidad que hay en bodega. Se desea sacar un informe (listado) con: número del producto, descripción, precio unitario, cantidad en bodega y valor de la existencia.

Los pasos que será necesario dar para obtener el listado pedido, si se piensa en un sistema mecánico de procesamiento de datos, son los siguientes:

a) La información contenida en el documento que llega de bodega se perfora en tarjetas y luego se verifica que haya sido perforada correctamente.

b) Las tarjetas obtenidas (tarjetas de detalle) se clasifican de acuerdo con el número del producto.

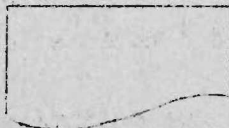
c) Las tarjetas clasificadas se intercalan con las del archivo maestro, seleccionando las tarjetas maestras sin tarjetas de detalle y las de detalle sin maestras. En este último caso se trata de un error que es necesario localizar y corregir.

Se obtiene un archivo ordenado por número de producto, en el que cada tarjeta maestra tiene a continuación su respectiva tarjeta de detalle.

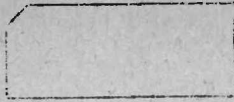
d) Se reproduce (gang-punch) la información, descripción y precio unitario de las tarjetas maestras en las de detalle.

e) Se seleccionan (separan) las tarjetas maestras de las de detalle

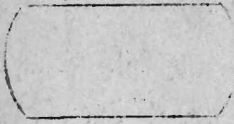
Para representar, mediante un diagrama de flujo, el proceso descrito, será necesario considerar las figuras que se indican a continuación:



para representar documentos e informes



para tarjetas perforadas



para operaciones efectuadas en máquinas con teclado



para representar arreglo en secuencia de un conjunto de ítem (SORT)



para representar la combinación de dos o más conjuntos de ítem en uno solo (MERGE)



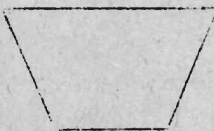
para extraer de un conjunto uno o más conjuntos específicos de ítem.



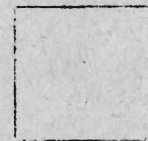
para representar la combinación con extracción; se forman dos o más conjuntos de ítem a partir de otros dos o más conjuntos (COLLATE)



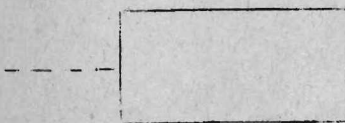
para archivos fuera de línea



para operaciones manuales que no requieran equipo



para operaciones de máquina suplementarias al procesamiento principal



para agregar comentarios descriptivos o notas aclaratorias

Con la ayuda de estos símbolos se construye el diagrama que se indica.

Ejemplo 12. Resolver el problema anterior con un sistema de procesamiento electrónico de datos (PED). Para ello debe considerarse que:

- a) La información registrada en el documento que llega de bodega ya ha sido perforada en tarjetas, pero no clasificada.
- b) El archivo maestro de productos ordenados por número, se tiene en cinta magnética.
- c) La información de las tarjetas de detalle se clasifica y se graba en disco magnético.

Los pasos que será necesario dar para obtener el informe pedido se indican a continuación:

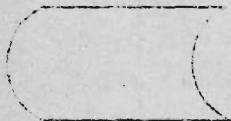
- i) Se leen las tarjetas de detalle y la información se almacena en la memoria del computador en forma de imagen de tarjeta.
- ii) La información leída se clasifica por número de producto y se graba en disco magnético.
- iii) Se leen registros del disco y de la cinta, se comparan y cuando corresponden al mismo producto, se efectúa el cálculo del valor de la existencia que hay en bodega.
- iv) Terminado el cálculo, se imprimen los resultados para obtener el informe pedido.
- v) Si hay registros en disco que no tienen su respectivo maestro en la cinta magnética, se imprimen para investigar el motivo.

Se necesitan dos nuevos símbolos para poder construir el diagrama de flujo.

Ellos son:

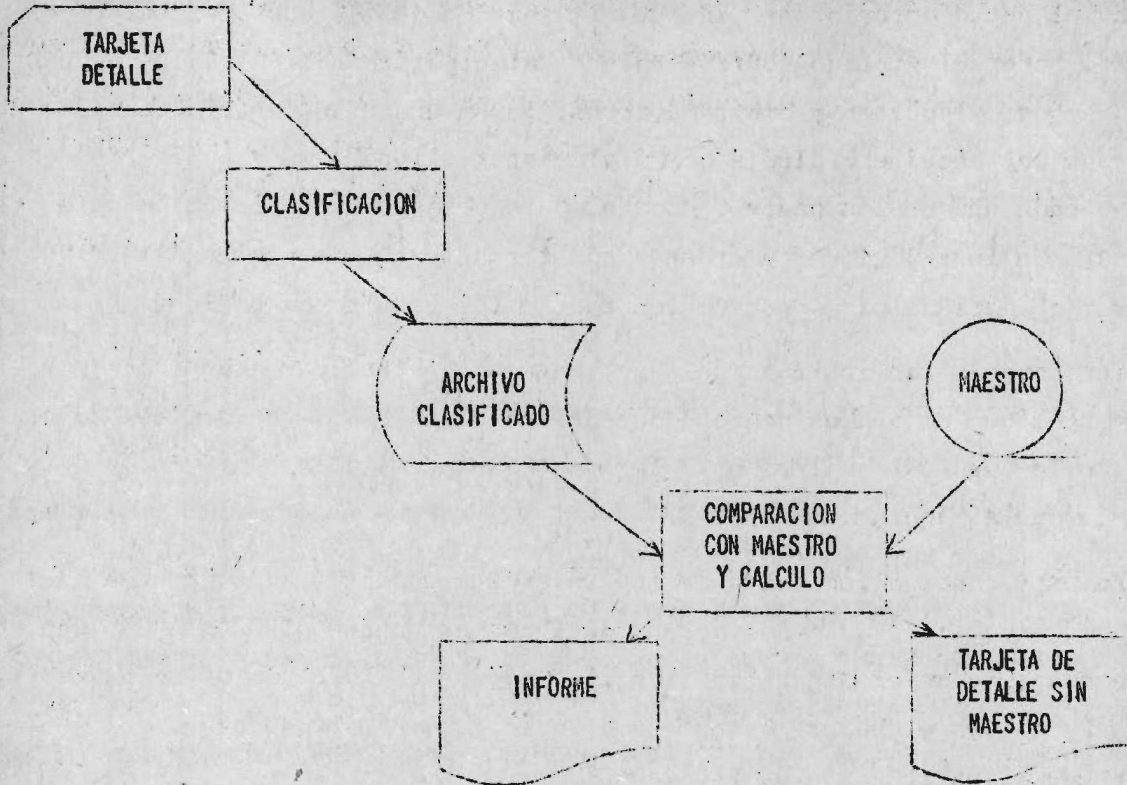


para representar cinta magnética en forma específica



para representar disco magnético, tambor magnético, etc.
Cualquier clase de almacenamiento en línea, utilizado para entrada/salida.

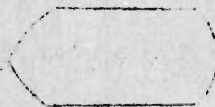
Se obtiene así el siguiente diagrama:



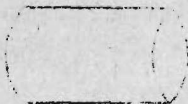
A continuación figuran otros símbolos que se utilizan;



para representar entrada de información manual por medio de máquina de teclado en línea.



para el despliegue de información a través de indicadores en línea, dispositivo de video, plotters, etc.



para representar tambor magnético en forma específica



para representar disco magnético en forma específica

III. ¿QUE ES UN PROGRAMA?

En el punto anterior se vió que el diagrama de flujo se construye con el objeto de hacer el gráfico correspondiente al algoritmo de solución de un problema. Ambos, algoritmo y diagrama, constituyen un medio de comunicación entre la persona que escribe o dibuja como solucionar el problema y quién tiene que ejecutar cada uno de los pasos indicados para obtener dicha solución. En otras palabras, son instrucciones u órdenes que forman parte de un lenguaje y que deben ser obedecidas para poder obtener el o los resultados pedidos.

Pero cuando las instrucciones no van a ser entregadas a una persona para que las realice, sino a un computador, es necesario buscar otro medio de comunicación, o lo que es lo mismo, otro lenguaje que permita la comprensión de ellas por parte de la máquina.

En este caso, al conjunto de instrucciones escritas que componen la solución se le denomina PROGRAMA y de ahí derivan los nombres PROGRAMADOR, que es la persona que traduce el algoritmo a un lenguaje entendible por el computador y PROGRAMAR, que es el nombre que se le da a la labor de traducción. El lenguaje a su vez se llama lenguaje de PROGRAMACION.

Es conveniente señalar que lo usual es que el algoritmo sea dado al programador sin pensar en que la solución va a ser obtenida mediante la utilización de un computador. De ese modo, el programador debe tener en cuenta todas las características del lenguaje y de la máquina que va a usar, con el objeto de que ellas le permitan obtener la solución más eficiente. Su labor, en consecuencia, es en gran parte creativa.

Es distinto el caso cuando el algoritmo entregado está desarrollado en detalle a nivel de operaciones elementales. Cuando eso ocurre, que no es lo habitual, la labor de programar se transforma en un trabajo de CODIFICACION, que deja muy poco margen al programador para que aplique su capacidad de creación.

El programa, como asimismo la información con la cual trabaja, están registrados en memoria. El computador debe "saber" en qué parte de ella ha sido almacenada la primera instrucción para poder así iniciar la ejecución del programa. La forma en que se comunica esa dirección dependerá del tipo de computador que se esté utilizando. Una vez que esto se ha logrado, las instrucciones mismas informarán al computador dónde debe ubicar los datos.

IV. LENGUAJES

Para comunicarse con el computador existen dos niveles de lenguaje:

1. Lenguaje de máquina

El lenguaje de máquina corresponde al nivel inferior y está formado por un conjunto de instrucciones elementales o básicas, diferente, por supuesto, de un computador a otro en lo que se refiere fundamentalmente al formato de las instrucciones, cuyo efecto y estructura dependerá de las características tecnológicas que tenga el computador y cuya escritura se hace a base, exclusivamente, de dígitos y pueden representar: operaciones, "direcciones" en memoria o parámetros en general. Esto implica que la lectura o revisión de un programa presente dificultades de interpretación a quien las haga, más aún, si la persona que cumple esa labor no está familiarizada con el computador utilizado o lo desconoce, le será muy difícil, si no imposible, conseguir su objetivo.

De acuerdo con las características del computador, la instrucción puede estar formada por:

- código de operación (OP)
- dirección del primer operando (D1)
- dirección del segundo operando (D2)
- dirección del resultado (D3)
- dirección de la próxima instrucción (D4)

lo que significa que podrá haber máquinas con instrucciones de una, dos, tres o cuatro direcciones. Además, en algunos computadores, no todas las instrucciones tienen igual longitud, de donde resultan máquinas con instrucciones de longitud fija y otras de longitud variable.

a) Se supondrá un computador ficticio cuya memoria está formada por 5000 celdas, cada una de ellas con capacidad para 18 dígitos. Se supone que el número 35 indica al computador que realice la operación SUMAR.

3500	35	1200	2400	3000	4000
	OP	D1	D2	D3	D4

En la celda 3500 se tiene una instrucción que indicará al computador lo siguiente: SUMAR al dato obtenido desde la celda 1200 el contenido de la celda 2400. Guardar el resultado en la celda 3000. La próxima instrucción se obtiene en la dirección 4000.

Todas las operaciones aritméticas se realizan en la unidad correspondiente, lo que permite hacer uso de los mismos datos todas las veces que se desee.

b) Se supondrá ahora que en el computador las operaciones son realizadas en un ACUMULADOR, siempre en la Unidad Aritmética, y que además, desde él, la transferencia de un resultado es automática. Sin embargo, es necesario cargar el acumulador con el primer operando, para lo cual se utilizará el código de operación 32.

3500	32	1200	0000	4000
	OP	D1	D3	D4
4000	35	2400	3000	4500
	OP	D1	D3	D4

En la celda 3500 se tiene una instrucción que le indicará al computador: LLEVAR el dato contenido en la celda 1200 al ACUMULADOR. La próxima instrucción se obtiene en la dirección 4000, y dice: SUMAR el contenido de la celda 2400 al contenido del acumulador, llevar el RESULTADO a la dirección 3000. La próxima instrucción se encuentra en la dirección 4500.

En la primera instrucción se puede observar que el campo D3 no se utiliza. Si cada uno de los campos ocupara una celda, la primera instrucción ocuparía tres celdas y la segunda cuatro. Se tendrían así instrucciones de longitud variable.

c) Se considerará ahora que la operación de llevar el resultado desde el acumulador a una celda se obtendrá mediante una nueva instrucción, en que el código de operación es 31.

3500	32	1200	4000
	OP	D1	D4
4000	35	2400	4500
	OP	D2	D4
4500	31	3000	5000
	OP	D3	D4

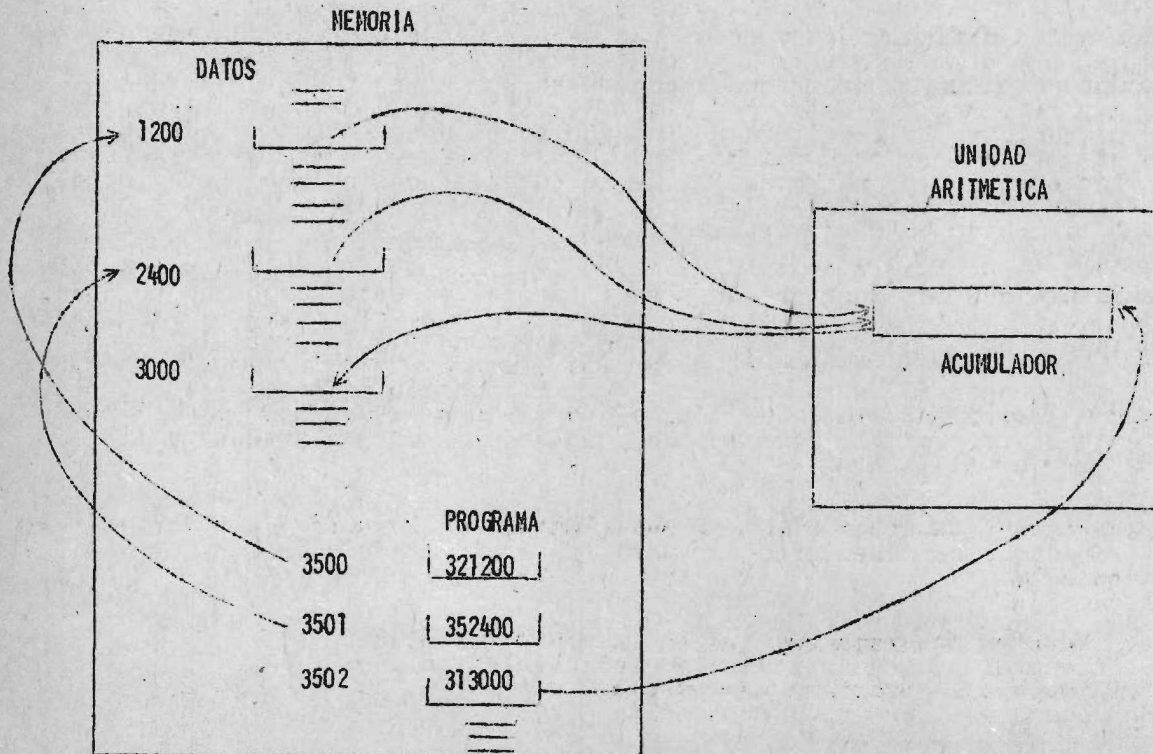
La instrucción contenida en la celda 3500 indica que el dato guardado en 1200 se lleva al acumulador y la próxima instrucción está en la dirección 4000.

En la celda 4000 la instrucción dice que al dato contenido en el acumulador se le suma el valor guardado en la celda 2400 y la próxima instrucción está en la dirección 4500.

Finalmente, en la celda 4500 se tiene: guardar en la celda 3000 el resultado que está en el acumulador, y buscar la próxima instrucción en la celda 5000.

d) Si se considera que todas las instrucciones se encuentran una a continuación de la otra en el mismo orden en que van a ser ejecutadas, es posible eliminar el campo correspondiente a la dirección de la próxima instrucción. Se obtiene así la estructura siguiente:

3500	32 1200
	OP D1
3501	35 2400
	OP D2
3502	31 3000
	OP D3



La instrucción contenida en la celda 3500 hace que el computador lleve al acumulador el dato que está en la dirección 1200. La instrucción siguiente hace que se sume el contenido de la celda 2400 a lo que hay en el acumulador. Por último, la instrucción contenida en la celda 3502 hace que se descargue el acumulador en la celda 3000.

Con los ejemplos anteriores es posible ver con claridad las dificultades que encuentra el programador o quien tenga que leer o revisar un programa escrito en lenguaje de máquina. Es más difícil aún si se utilizan en la escritura otros sistemas numéricos.

Sólo con el ánimo de dar un punto de referencia más, se expone a continuación el mismo problema resuelto en los ejemplos, pero ahora utilizando el lenguaje de un computador real, un IBM/360. No interesa el análisis de cada instrucción, sino el aspecto que presenta el conjunto de ellas.

58	50	20	OA
58	70	20	OE
1A	57		
50	50	20	06

2. Lenguaje simbólico

Debido a las dificultades mencionadas, fue necesario crear los lenguajes simbólicos que permiten al programador concentrarse más en la solución del problema y no tanto en la escritura de las instrucciones. Al mismo tiempo se disminuyen los errores de programación, de lectura, de perforación, etc., como, asimismo, el tiempo de búsqueda de ellos.

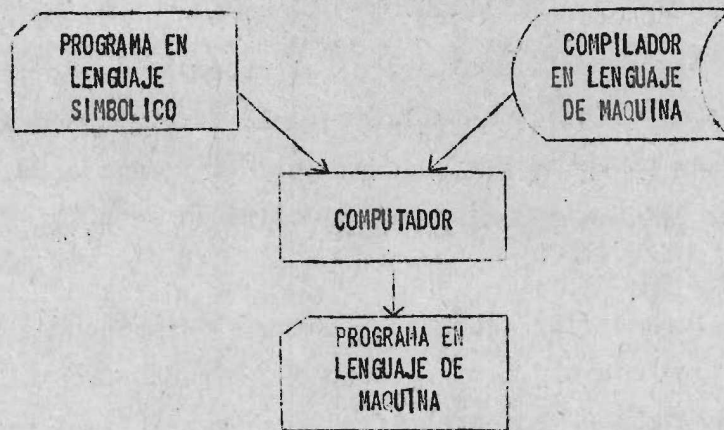
Es evidente que las instrucciones escritas en lenguaje simbólico no podrán ser "entendidas" por el computador, dado que este conoce solamente el lenguaje de máquina. Es necesario, entonces, hacer una traducción de las instrucciones de un lenguaje a otro para lo cual se hace uso de un programa traductor llamado COMPILADOR.

Las etapas que se deben efectuar para lograr la traducción son las indicadas a continuación:

- Se tiene el programa escrito en lenguaje simbólico
- Se almacena el programa en memoria

- Se almacena en memoria, el compilador, que está escrito en lenguaje de máquina
- Se realiza el proceso de traducción
- Como resultado, se obtiene el programa traducido, esto es, en lenguaje de máquina.

Para "graficar" este proceso se supondrá que el programa escrito en lenguaje simbólico se ha perforado en tarjetas y que el compilador está grabado en disco magnético.



Entre los lenguajes simbólicos es necesario distinguir tres categorías:

A. Los orientados a la máquina, en los que los elementos que conforman una instrucción de máquina han sido reemplazados en su totalidad, o en forma parcial, por símbolos. Dado que el conjunto de instrucciones de máquina y la estructura de éstas dependen de las características del computador, el lenguaje simbólico que considere directa o indirectamente esas características estará orientado a él. El programa que se vio escrito en lenguaje de máquinas del IBM/360, al ser escrito en el lenguaje simbólico orientado a ese computador, quedará en la siguiente forma:

```
L 5,DAT01
L 7,DAT02
AR 5,7
ST 5,RESULT
```

Este lenguaje específico se llama ASSEMBLER y corresponde al tipo conocido como ensamblador, en que cada instrucción del lenguaje da origen a una sola instrucción de máquina. Esto significa que los algoritmos escritos en un lenguaje ensamblador resultan tan extensos como los escritos en lenguaje de máquina.

Además, aun cuando la estructura de las instrucciones es más simple que en lenguaje de máquina y sus símbolos, fáciles de recordar y revisar, todavía no ofrecen mucha información a quien no esté interiorizado en las convenciones de su escritura.

B. Los lenguajes generales o de alto nivel permiten escribir los algoritmos en una forma fácil de comprender pues se aproximan bastante al lenguaje que se habla a diario. Esto se debe a que las expresiones son similares a las que se escriben en matemática elemental, a que los nombres con que se designan las variables pueden ser tan descriptivos como se desee, y a que es posible colocar comentarios acerca de lo que realiza el programa, parte de programa o instrucción, en el idioma que la persona quiera, castellano, inglés u otro. Además, cada instrucción del lenguaje de alto nivel corresponde a varias instrucciones en lenguaje de máquina. Por ello es que las primeras se denominan sentencias, declaraciones o proposiciones.

Para lograr lo anterior todos los lenguajes disponen de constantes numéricas que se escriben con o sin signo, con o sin punto decimal (en reemplazo de la coma decimal). También, todos permiten usar variables y todos exigen que los nombres que identifican variables comiencen por carácter alfabético. Hay diferencia, sí, entre un lenguaje y otro en la cantidad de caracteres que componen el nombre. Por ejemplo, en BASIC puede tener hasta dos caracteres, en FORTRAN hasta seis; en cambio PL/I y COBOL aceptan hasta treinta caracteres.

En todos los lenguajes es posible "romper" la secuencia normal de ejecución mediante sentencias de bifurcación o de "salto" de una sentencia a otra que puede estar antes o después en el programa. Estos saltos pueden ser de acuerdo con una condición o incondicionales y, en este caso, la sentencia es del tipo

GO TO X

en que X es un rótulo o etiqueta que identifica a otra sentencia que es la meta del salto. En BASIC todas las sentencias deben llevar rótulos numéricos; en FORTRAN los rótulos deben ser numéricos pero no es obligatorio que todas las sentencias lo tengan; en PL/I y COBOL los rótulos son alfanuméricos y se utilizan igual que en FORTRAN.

Los saltos condicionales pueden estar complementados con sentencias IF que tienen una estructura distinta según sea el lenguaje al que pertenecen.

Dado que un computador sólo puede realizar las operaciones aritméticas, todos los lenguajes permiten formar expresiones aritméticas cualquiera que sea su complejidad. Las variables que se utilicen pueden ser simples o con índices. Si se trata de estas últimas, la única exigencia de los lenguajes es que las dimensiones de los arreglos sean "declaradas" al comienzo del programa. PL/I difiere de los otros tres lenguajes en que permite usar índices cero o negativos si es necesario, no así los otros.

Ejemplo 13. Problema de la tabla estado civil-edad. En las páginas siguientes se verá, con los cuatro lenguajes mencionados, la solución del ejemplo 8, obtención de una tabla estado civil por edad. Las soluciones están escritas en Hojas de Codificación y se ha cuidado de diferenciar la letra O del número 0 escribiendo una barra diagonal sobre la letra. Cada línea de la hoja corresponde a una tarjeta.

1. Solución en el lenguaje FORTRAN

Esta solución está escrita en una variante de FORTRAN llamada WATFOR (un FORTRAN desarrollado en la Universidad de WATERLOO), utilizada con muy buenos resultados con propósitos educativos. Se diferencia del FORTRAN principalmente en las sentencias de entrada/salida de datos.

Las sentencias de comentario empiezan con la letra C en la primera columna. Los rótulos se codifican en las columnas 1 a 5 y el texto de la sentencia en las columnas 7 a 72.

La declaración de las dimensiones del arreglo M se hace por medio de la sentencia DIMENSION, ubicada al comienzo del programa.

Cuando se detecta fin de datos en la operación de lectura, se produce un salto automático a la parte del programa destinada a la impresión de los resultados. Esto se obtiene mediante la cláusula END=40 colocada en la sentencia READ.

SOLUCIÓN FORTRAN

```
C
C SOLUCIÓN FORTRAN: TABULACIÓN DE EDAD, ESTADO CIVIL
C
C     DIMENSIÓN M(100,7)
C
C 1.- INICIALIZAR M CON CEROS:
C
C     J=1
C     1 I=1
C     2 M(I,J)=0
C       I=I+1
C       IF(I.LE.100) GØ TØ 2
C       J=J+1
C       IF(J.LE.7) GØ TØ 1
C
C 2.- LECTURA Y CONTABILIZACIÓN DEL CASO:
C
C     3 READ (END=40) EDAD,ECIV
C       M(EDAD,ECIV)=M(EDAD,ECIV)+1
C       GØ TØ 3
C
C 3.- IMPRESIÓN DE LA MATRIZ:
C
C     40 I=1
C     4 PRINT, (M(I,J),J=1,7)
C       I=I+1
C       IF(I.LE.100) GØ TØ 4
C
C     STØP
C
C LA SENTENCIA END INDICA AL COMPILADOR QUE NO HAY
C MAS SENTENCIAS POR TRADUCIR:
C
C     END
```


2. Solución en el lenguaje PL/I

Está escrita usando algunas opciones básicas del lenguaje. En PL/I cada sentencia termina con punto y coma (;) y pueden escribirse varias sentencias en una tarjeta. Se puede codificar desde la columna 2 a la 72, en forma absolutamente libre. Los comentarios se escriben precedidos por los caracteres barra diagonal y asterisco (/*) y seguidos por asterisco, barra diagonal (*/):

```
/* ESTE ES UN COMENTARIO. */
```

Los rótulos se distinguen del texto de la sentencia porque se separan de ella con el signo dos puntos (:).

Las dimensiones de los arreglos se declaran mediante una sentencia DECLARE.

Para "inicializar" un arreglo con ceros, basta asignarle ceros al nombre del arreglo, sin necesidad de especificar índices.

La sentencia ON ENDFILE (SYSIN) indica lo que debe hacerse cuando se terminan los datos.

La sentencia END indica al compilador que no hay más sentencias por traducir y además indica término del proceso.

SOLUCIØN PL/I

/*

- SOLUCIØN PL/I: TABULACIØN DE EDAD, ESTADØ CIVIL. */

TABULA: PRØCEDURE ØPTIONS(MAIN);

DECLARE M(100,7);

/* 1.- INICIALIZA M CØN CERØS: */

M=0;

/* 2.- LECTURA Y CØNTABILIZACIØN: */

ØN ENDFILE (SYSIN) GØ TØ IMPRIMIR;

LEER; GET FILE (SYSIN) LIST (EDAD,ECIV);

M(EDAD,ECIV)=M(EDAD,ECIV) +1; GØ TØ LEER;

/* 3.- IMPRIME MATRIZ M: */

IMPRIMIR: I=1;

IMPRIME: PUT LIST (M(I,J) DØ J=1 TØ 7);

I=I+1; IF I<= 100 THEN GØ TØ IMPRIME;

/*

LA SENTENCIA END DEFINE EL FIN DE LA EJECUCIØN
Y EL FIN DE LAS SENTENCIAS A LA VEZ:

*/

END;

3. Solución en el lenguaje BASIC

BASIC es un lenguaje orientado al trabajo de terminales de tipo interactivo. Todas las sentencias de un programa escrito en BASIC deben llevar un rótulo, que es el número de secuencia de la misma. Se acostumbra numerar las sentencias de 10 en 10 para facilitar la intercalación de otras nuevas, entre las ya definidas. El programa se escribe en una máquina de teclado, conectada a un computador y en ella aparecen los diagnósticos de error. El tiempo de respuesta comunicándolos es muy corto, lo que permite corregirlos inmediatamente.

La lectura de datos, hecha mediante una sentencia INPUT, no contempla la posibilidad de detectar el fin de éstos. La sentencia INPUT solicita, cada vez que es ejecutada, que los datos se escriban en el teclado de la máquina y no tiene posibilidad de estipular que se ejecute algún fragmento distinto de programa al término de ellos. Por esto, en la adición se ha hecho uso de un truco de programación: la edad 9999999 indicará fin de datos. Al ser detectada esa "marca" se imprimirá la matriz.

Los comentarios se inician con la clave REM (de REMARKS). La declaración de las dimensiones de un arreglo se hace con una sentencia DIM (de DIMENSION). Las asignaciones se efectúan con la sentencia LET por ejemplo:

```
LET A=0
LET H=N
```

La impresión de arreglos se realiza con la sentencia MAT PRINT y la escritura se obtiene de renglón a renglón en la máquina de escribir.

Finalmente, en la solución se han utilizado las variables E1 y E2 para identificar la edad y el estado civil, respectivamente.

SOLUCIØN BASIC

```
10 REM
20 REM     SOLUCIØN BASIC: TABULACIØN DE EDAD, ESTADØ CIVIL.
30 REM
40 REM
50 DIM M(100,7)
60 REM
70 REM 1.- INICIALIZAR M CØN CERØS:
80 REM
90 LET J=1
100 LET I=1
110 LET M(I,J)=0
120 LET I=I+1
130 IF I<= 100 THEN 110
140 LET J=J+1
150 IF J<= 7 THEN 100
160 REM
170 REM 2.- LECTURA Y CØNTABILIZACIØN:
180 REM
190 INPUT E1,E2
200 IF E1 = 9999999 THEN 260
210 LET M(E1,E2) = M(E1,E2) + 1
220 GØ TØ 190
230 REM
240 REM 3.- IMPRESIØN DE LA MATRIZ M:
250 REM
260 MAT PRINT M
270 END
```

Para que el programa se ejecute, una vez que se han dado todas las sentencias a través de la máquina de escribir, es necesario escribir a máquina la sentencia especial

RUN

que no es parte del programa, sino que pide al computador la ejecución de éste.

4. Solución en el lenguaje COBOL

COBOL es un lenguaje apropiado para resolver problemas del tipo llamado "comercial", dado que tiene una estructura sintáctica que permite hacer programas muy narrativos.

Está formado por cuatro divisiones, cada una de las cuales tiene una función determinada que cumplir. Algunas de ellas están compuestas por secciones.

Un esqueleto de programa tendrá la estructura siguiente:

IDENTIFICATION DIVISION

Fundamentalmente para identificar el programa.

ENVIRONMENT DIVISION

Para indicar el medio ambiente en que debe trabajar, computador, unidades lógicas, etc.

DATA DIVISION

Para describir archivos, registros, y toda clase de variables que se utilicen en el programa.

PROCEDURE DIVISION

Contiene el algoritmo de solución del problema, traducido a sentencias del lenguaje.

Para insertar comentarios debe ponerse un asterisco (*) en la columna 7. Los nombres de división, sección, etc. e identificadores de sentencias se escriben a partir de la columna 8 y el cuerpo de las sentencias en las columnas 12 a la 72.

SOLUCION COBOL

IDENTIFICATION DIVISION.

PROGRAM-ID. 'TABULA'.

REMARKS. TABULACION DE EDAD,ESTADO CIVIL.

ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.

SELECT TARJETA ASSIGN TO 'SYSIN' UTILITY.

SELECT LISTADO ASSIGN TO 'SYSOUT' UTILITY.

DATA DIVISION.

FILE SECTION.

FD TARJETA RECORDING MODE IS F
 LABEL RECORDS ARE OMITTED
 DATA RECORD IS DATOS.

01 DATOS.

05 EDAD PICTURE 99.

05 ECIV PICTURE 9.

05 FILLER PIC X(78).

FD LISTADO RECORDING MODE IS F
 LABEL RECORD IS OMITTED
 DATA RECORD IS LINEA.

01 LINEA.

05 FILLER PICTURE X(132).

WORKING-STORAGE SECTION.

77 I PICTURE 9(3) VALUE ZEROS.

77 J PICTURE 9 VALUE ZEROS.

01 MATRIZ.

05 EDADES OCCURS 100 TIMES.

10 ECIVIL OCCURS 7 TIMES.

15 M PICTURE 9(6).

01 LINEA1.

05 FILLER OCCURS 6 TIMES.

10 CELDA PICTURE ZZZZZ9.

PROCEDURE DIVISION.

INICIACION-DEL-TRABAJO SECTION.

INICIA.

MOVE 1 TO J.

DEJA-I-EN-UNO.

MOVE 1 TO I.

CARGA. MOVE ZERO TO M (I, J) ADD 1 TO I

IF I IS LESS OR EQUAL TO 100 THEN GO TO CARGA.

ADD 1 TO J

IF I IS NOT GREATER 7 THEN GO TO DEJA-I-EN-UNO.

OPEN INPUT TARJETA

OUTPUT LISTADO.

LECTURA-Y-CONTABILIZACION SECTION.

LEE.

READ TARJETA AT END GO TO FIN.
 ADD 1 TO M (EDAD, ECIV)
 GO TO LEE.

IMPRESION-DE-RESULTADOS SECTION.

FIN.

MOVE 1 TO I.

PREPARA.

MOVE 1 TO J.

MUEVE.

MOVE M (I, J) TO CELDA (J) ADD 1 TO J
 IF J IS NOT GREATER 7 THEN GO TO MUEVE.

WRITE LINEA FROM LINEA1 AFTER 1.

IF I IS NOT GREATER 100 THEN GO TO PREPARA.

CIERR ARCHIVOS.

CLOSE TARJETA.

CLOSE LISTADO.

STOP RUN.

C. Por último, los lenguajes orientados al problema: El uso de los lenguajes de tipo general es recomendable cuando las necesidades de procesamiento de datos caen dentro de un amplio rango de aplicaciones.

En cambio cuando las aplicaciones son mucho más circunscritas y corresponden a un único tipo de problemas, se pueden reducir radicalmente los esfuerzos de programación con un programa que permite abarcar muchos casos parecidos y que tenga, además, una forma simple de introducir los parámetros específicos del problema. Este tipo de programas, independientemente del lenguaje en que se hayan escrito, se conocen como programas orientados, con lo que se quiere expresar que son aptos para resolver un determinado tipo de problemas.

Cuando las funciones de los programas orientados son demasiado complejos, puede hacerse difícil entregar los parámetros del problema. Este es el caso de programas orientados a simulación, a resolución de modelos lineales, al cálculo de estructuras, a la producción de tabulaciones, al análisis estadístico, etc. En estas oportunidades los parámetros son tantos que se prefiere crear lenguajes orientados, que facilitan la comunicación entre el usuario y el programa orientado.

Los lenguajes orientados, generalmente, siguen reglas parecidas a las de los lenguajes de alto nivel. Es frecuente que dispongan de variables, de constantes, de sentencias condicionales y de ciertos recursos aritméticos. No obstante lo

anterior, no debe confundirse el concepto de lenguaje de alto nivel con el de lenguaje orientado. En tanto el primero sirve a cualquier propósito y es traducido siempre a instrucciones de lenguaje de máquina para la ejecución de los programas del usuario, cada lenguaje orientado permite sólo la realización de tareas dentro de ámbitos específicos y generalmente sólo permite expresar de una manera más cómoda los parámetros de un problema que será resuelto por el programa orientado.

CELADE, utiliza, en el manejo de problemas estadísticos, procedimientos y programas orientados de excepcional utilidad. Entre ellos, los programas:

MARGINAL	orientado a la inspección estadística y evaluación de códigos de archivos
CENTS	orientado a la producción de tabulaciones en censos y encuestas de gran tamaño
SPSS y OSIRIS	orientados al análisis estadístico de encuestas.

El primero no posee un lenguaje orientado que lo maneje, de tal manera que es necesario entregarle las características de los archivos a través de tarjetas de control. En cambio, los restantes poseen poderosos lenguajes orientados que ofrecen una gama mayor de posibilidades de programación. Estos lenguajes se denominan con los mismos nombres de los programas.

CELADE ha desarrollado un programa, dotado de lenguaje orientado, el CONCOR, para cubrir el área de conversión y corrección automática de errores para masas de datos estadísticos.

V. ESTRUCTURACION DE PROGRAMAS Y MODULARIDAD

La estructuración o diseño de programas es, tal vez, el concepto más importante que debe tener siempre presente el programador. Una buena estructuración implica mayor posibilidad de funcionamiento del programa como asimismo un costo menor. Por otra parte, los programas bien diseñados son más fáciles de escribir, documentar, revisar, cambiar, entender, depurar, probar y mantener.

¿Qué es lo que hace que un programa esté bien estructurado? La respuesta, si bien es simple, es un desafío a la capacidad creadora del programador; un programa está bien estructurado si consta de dos tipos de rutinas:

- i) Monitores o módulos de control.
- ii) Subrutinas, cada una de las cuales realiza una función bien específica, parte de la función total del programa.

Esta definición encierra un concepto nuevo, que es el de modularidad entendiéndose por tal la estructuración de un programa en módulos y llamando módulo a un grupo de instrucciones que realizan una función claramente definida, específicamente relacionada a la función lógica del programa.

Profundizando lo anterior, se tiene que: el módulo de control debe dar una visión rápida acerca de la función total del programa. Para ello debe estar constituido fundamentalmente por llamadas a las subrutinas, que son módulos que cumplen con tareas particulares.

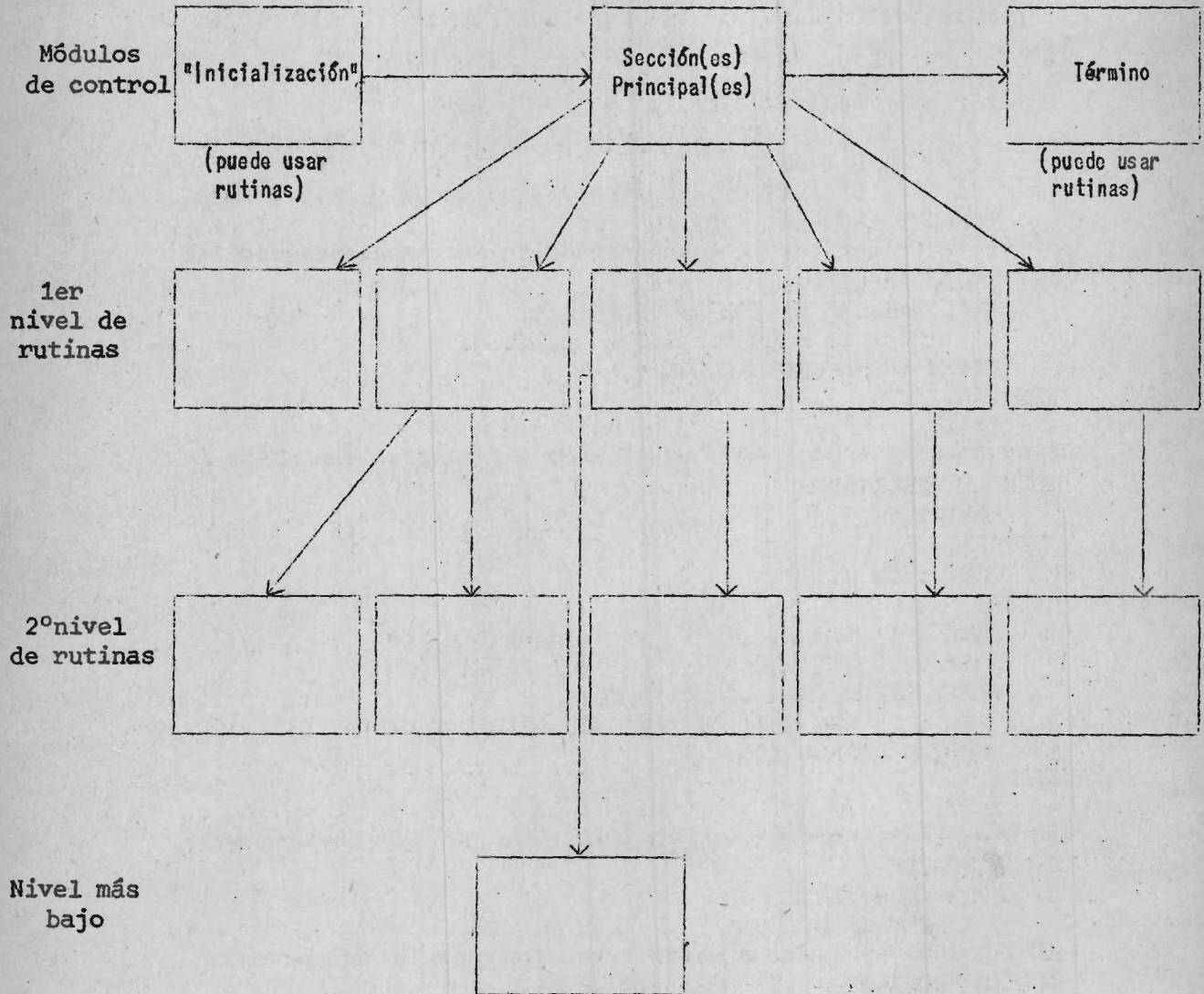
Se puede decir que el módulo de control está ubicado en el primer nivel de jerarquía en la estructura del programa y en su diseño es donde debe concentrar su esfuerzo el programador. Aquí es donde podrá ver claramente el panorama total del programa y por ello en esta etapa no debe entrar al detalle. Se recomienda que el diagrama de bloques correspondiente al módulo de control no exceda de los veinte bloques. Esto tiene como ventaja el poder repensar y rediagramar fácilmente el programa.

Las subrutinas en cuanto a estructura, son modelos del programa, así como éste lo es del sistema completo. Esto significa que la subrutina también puede consistir en un módulo de control en el que se hacen llamados a módulos de un nivel más bajo.

Però es importante dejar en claro o insistir en el hecho de que modularidad no significa hacer uso de rutinas porque sí. Estas deben cumplir ciertos requisitos. Se dijo anteriormente que la rutina tiene una función específica, parte de la función lógica total. Hay que agregar que no debe haber, o debe haber muy

poca, interacción con otros módulos; en otras palabras, no es dependiente de lo que ocurra en ellos, sino solamente de la información que se le entregue y de la función que cumpla. Esto significa que puede ser revisado y especificado en detalle sin considerar al resto, lo que trae como consecuencia que cada módulo podría ser programado por personas distintas. Finalmente, no deben entregarse a rutinas funciones que corresponden al módulo de control.

Recurriendo al diagrama de bloques, se puede representar claramente la estructura de un programa diseñado con el concepto de modularidad.



Es probable que el problema que se resolvió con FORTRAN BASIC PL/I y COBOL no sea el mejor ejemplo para mostrar la aplicación del concepto de modularidad, sin embargo, se ha tomado la solución COBOL y de ella la división de procedimiento para enfocar el mismo problema con la ayuda de las nuevas ideas sobre estructura de programas.

SOLUCIØN CØBØL

PRØCEDURE DIVISIØN.

PERFØRM INICIA-Y-DEJA-CERØ-EN-M THRU LEE-Y-CØNTABILIZA.
PERFØRM LEE-Y-CØNTABILIZA THRU TERMINA.
PERFØRM IMPRIME-RESULTADØS THRU FUERA.
GØ TØ FIN.

INICIA-Y-DEJA-CERØ-EN-M.

MØVE 1 TØ J.

DEJA-I-EN-UNØ.

MØVE 1 TØ I.

CARGA. MØVE ZERØ TØ M (I, J) ADD 1 TØ I

IF I IS LESS ØR EQUAL TØ 100 THEN GØ TØ CARGA.

ADD 1 TØ J

IF I IS NOT GREATER 7 THEN GØ TØ DEJA-I-EN-UNØ.

PERFØRM ABRE-ARCHIVØS.

LEE-Y-CØNTABILIZA.

READ TARJETA AT END GØ TØ TERMINA.

ADD 1 TØ M (EDAD, ECIV)

GØ TØ LEE-Y-CØNTABILIZA.

TERMINA.

EXIT.

IMPRIME-RESULTADØS.

MØVE 1 TØ I.

PREPARA.

MØVE 1 TØ J.

MUEVE.

MØVE M (I, J) TØ CELDA (J) ADD 1 TØ J

IF J IS NOT GREATER 7 THEN GØ TØ MUEVE.

WRITE LINEA FROM LINEA1 AFTER 1

IF I IS NOT GREATER 100 THEN GØ TØ PREPARA.

PERFØRM CIERRA-ARCHIVØS.

FUERA.

EXIT.

ABRE-ARCHIVØS.

ØPEN INPUT TARJETA

ØUTPUT LISTADØ.

CIERRA-ARCHIVØS.

CLØSE TARJETA.

CLØSE LISTADØ.

FIN.

STØP RUN.

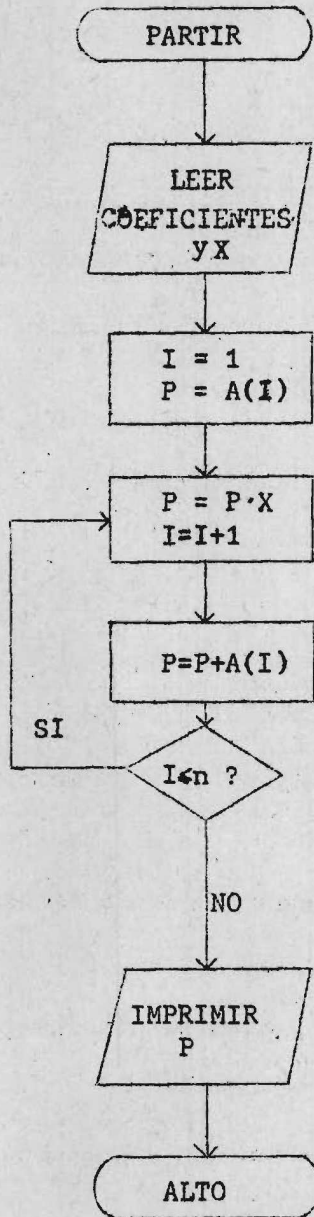
VI. PROBLEMAS RESUELTOS

1. Evaluar el polinomio de orden n

$$P = a_1x^n + a_2x^{n-1} + \dots + a_nx + a_{n+1}$$

Solución: El polinomio

$$P = (((a_1x+a_2)x+a_3)x+ \dots +a_n)x+a_{n+1}$$



2. Encontrar todos los números de tres dígitos que sean iguales a la suma de los cubos de sus dígitos.

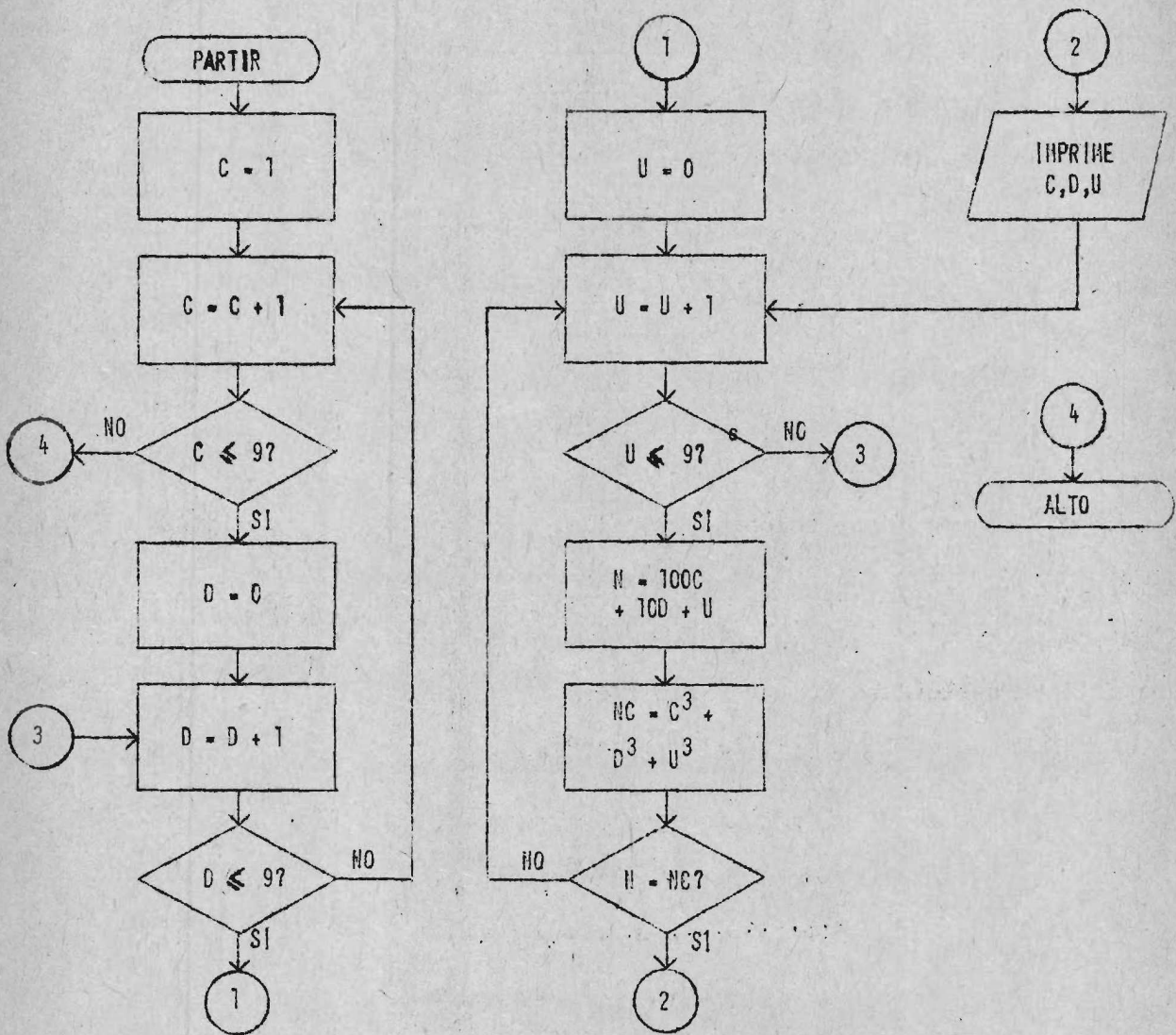
Solución: Si se denominan C, D y U a los dígitos del número, el número será:

$$N=100C+10D+U$$

por otra parte, la suma de los cubos de los dígitos corresponde a:

$$NC=C^3+D^3+U^3$$

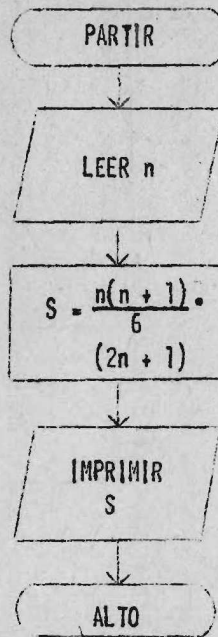
se trata de obtener que: N sea igual a NC



3. Encontrar la suma de los cuadrados de los primeros 101 enteros positivos mediante la fórmula general

$$S = \frac{n(n+1)(2n+1)}{6}$$

donde n es el número de término

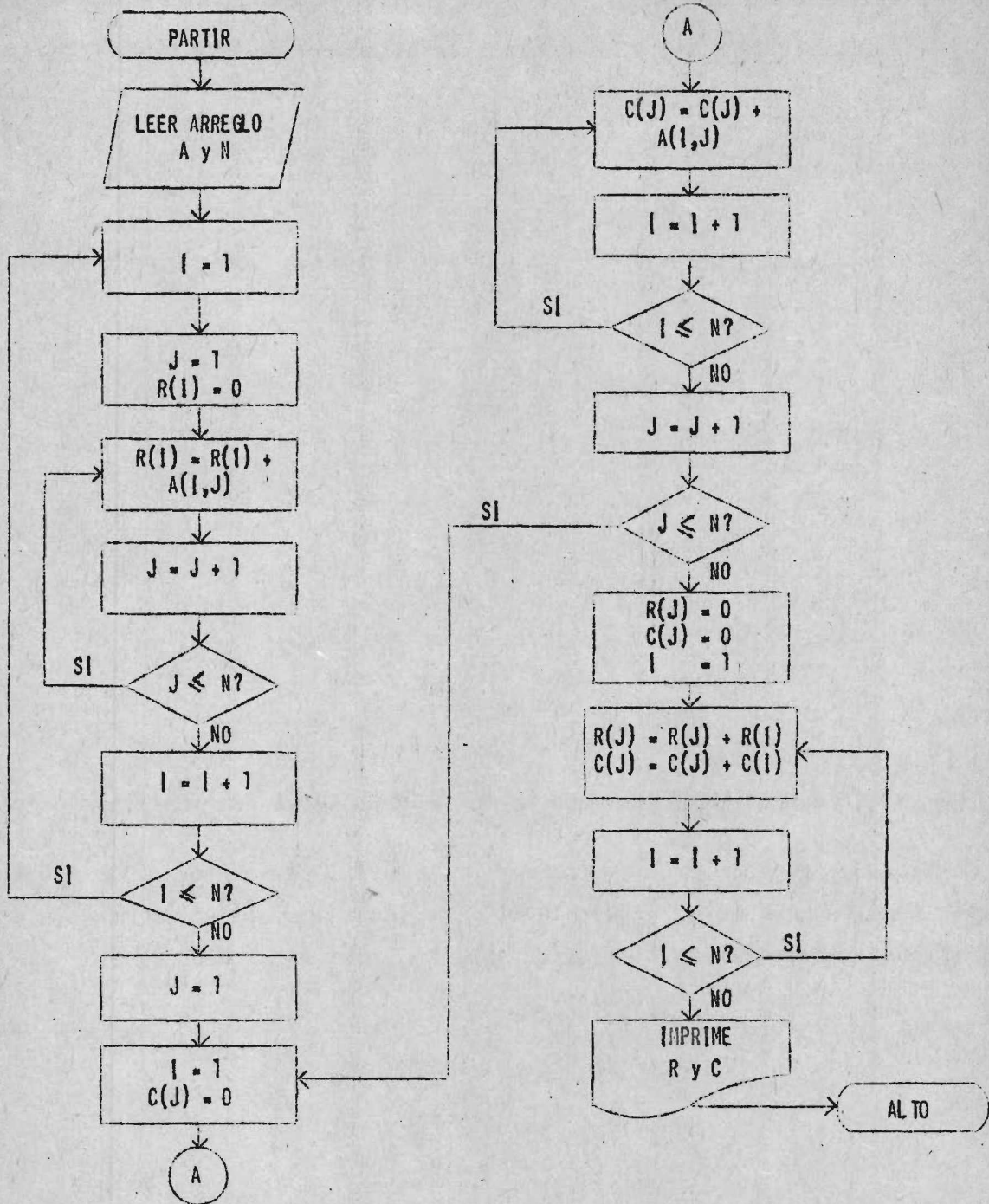


Compárese con la solución dada en el ejemplo 4 de este capítulo.

4. Se tiene un arreglo A de $n \times n$ elementos. Se pide sumar todos los elementos de cada renglón para obtener subtotales R_i y todos los elementos de cada columna para obtener subtotales C_j . Verificar que R_{n+1} sea igual a C_{n+1} .

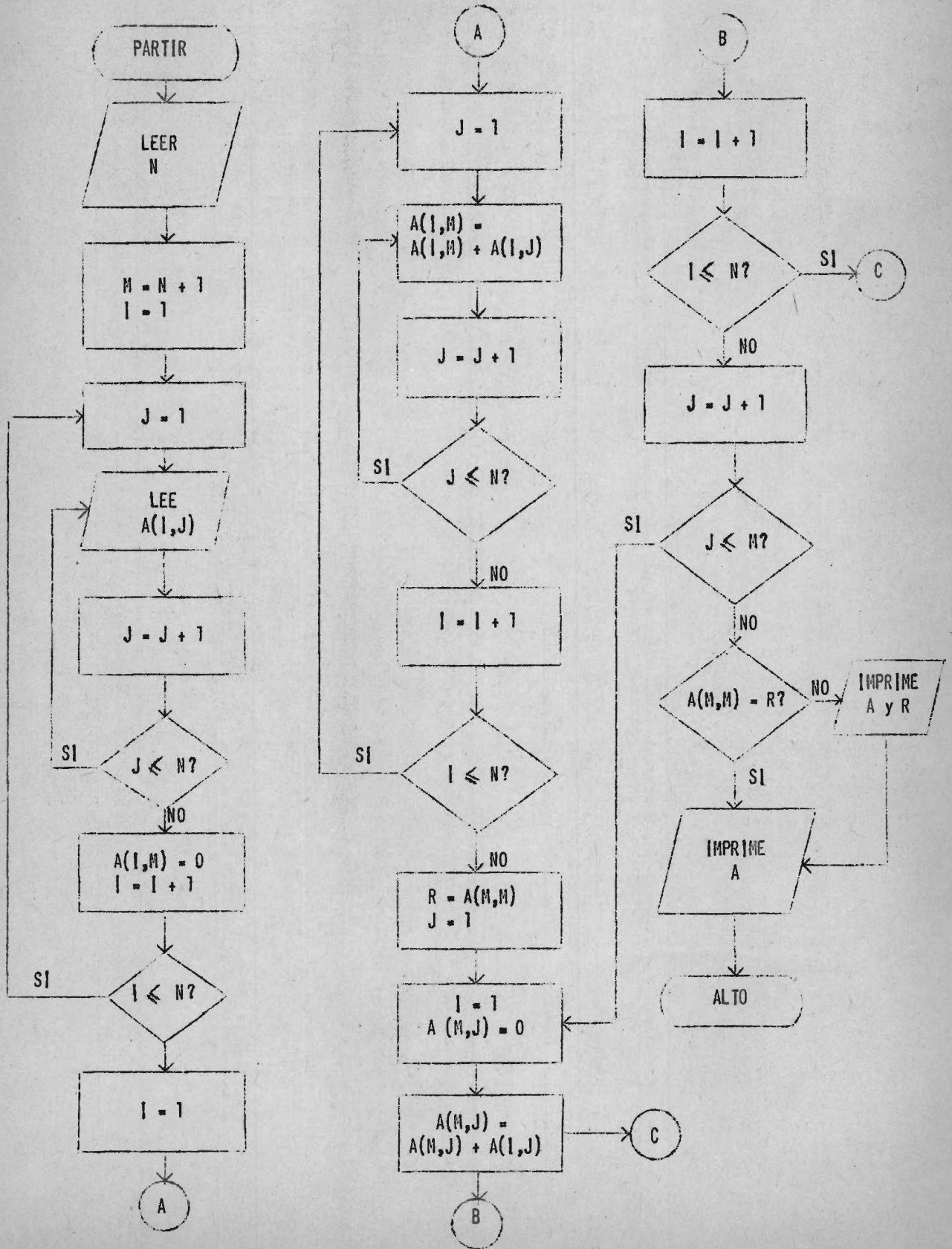
a_{11}	a_{12}	a_{13}	\dots	a_{1n}	R_1
a_{21}	a_{22}	a_{23}	\dots	a_{2n}	R_2
a_{31}	a_{32}	a_{33}	\dots	a_{3n}	R_3
.....					
a_{n1}	a_{n2}	a_{n3}	\dots	a_{nn}	R_n
C_1	C_2	C_3	\dots	C_n	R_{n+1} C_{n+1}

a) Solución en que los arreglos R y C se construyen aparte de A.



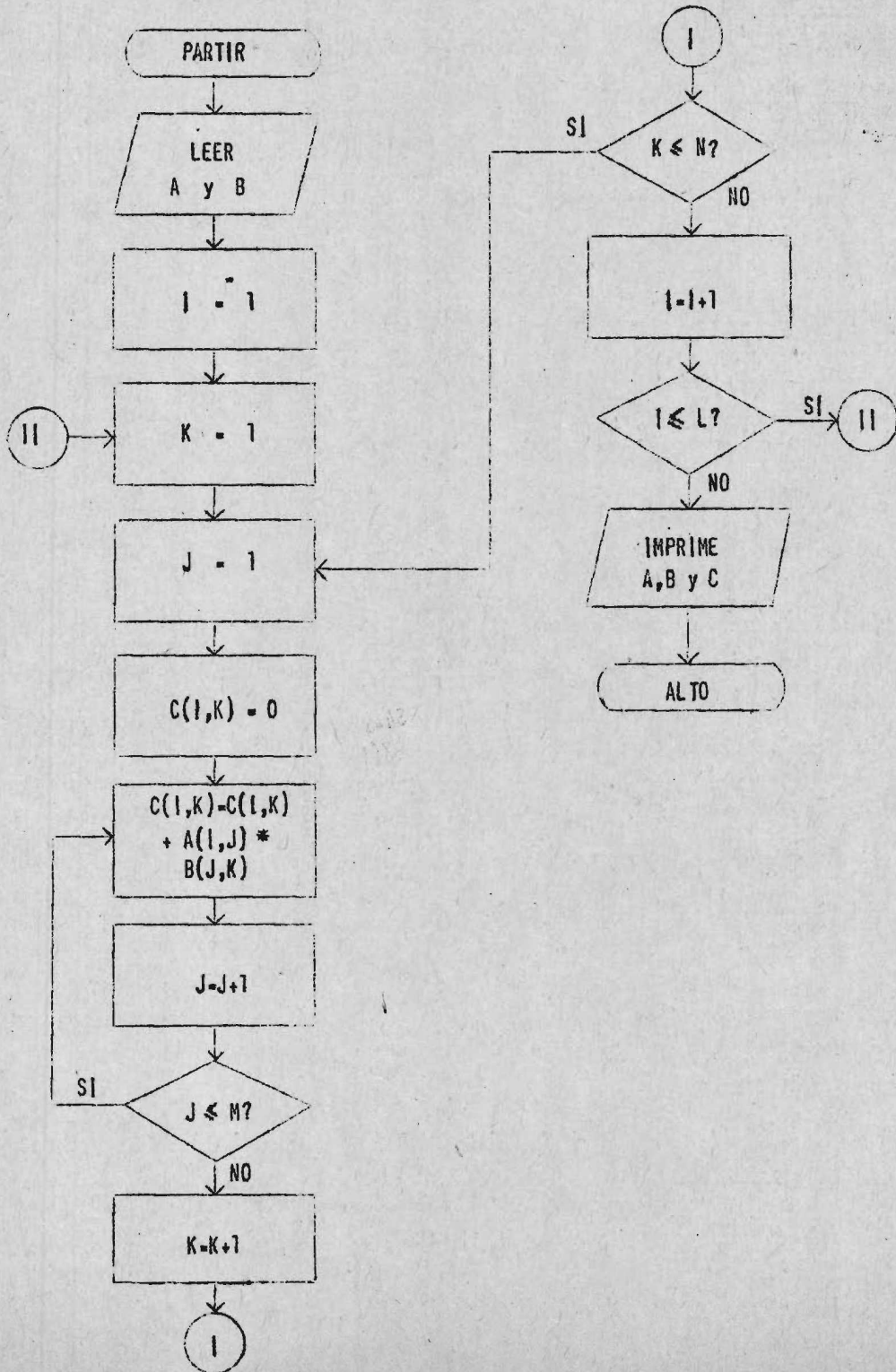
En esta solución no se ha incluido la verificación de la igualdad de R_{n+1} y C_{n+1} dado que ella se puede realizar visualmente con el resultado de la impresión.

b) Solución en que los subtotales por renglón y por columna son elementos de un arreglo A aumentado.



5. Calcular el producto de dos arreglos A(16x20) y B(20x25) a base de la fórmula del elemento general de la matriz resultante C:

$$c_{i,k} = \sum_{j=1}^n a_{i,j} b_{j,k}$$



6. Se tiene una serie de N puntos $(x_1, Y_1), (X_2, Y_2) \dots (X_n, Y_n)$. Se pide encontrar la ecuación de la recta que pasa por los puntos. La ecuación de la recta es:

$$Y = A_0 + A_1 X$$

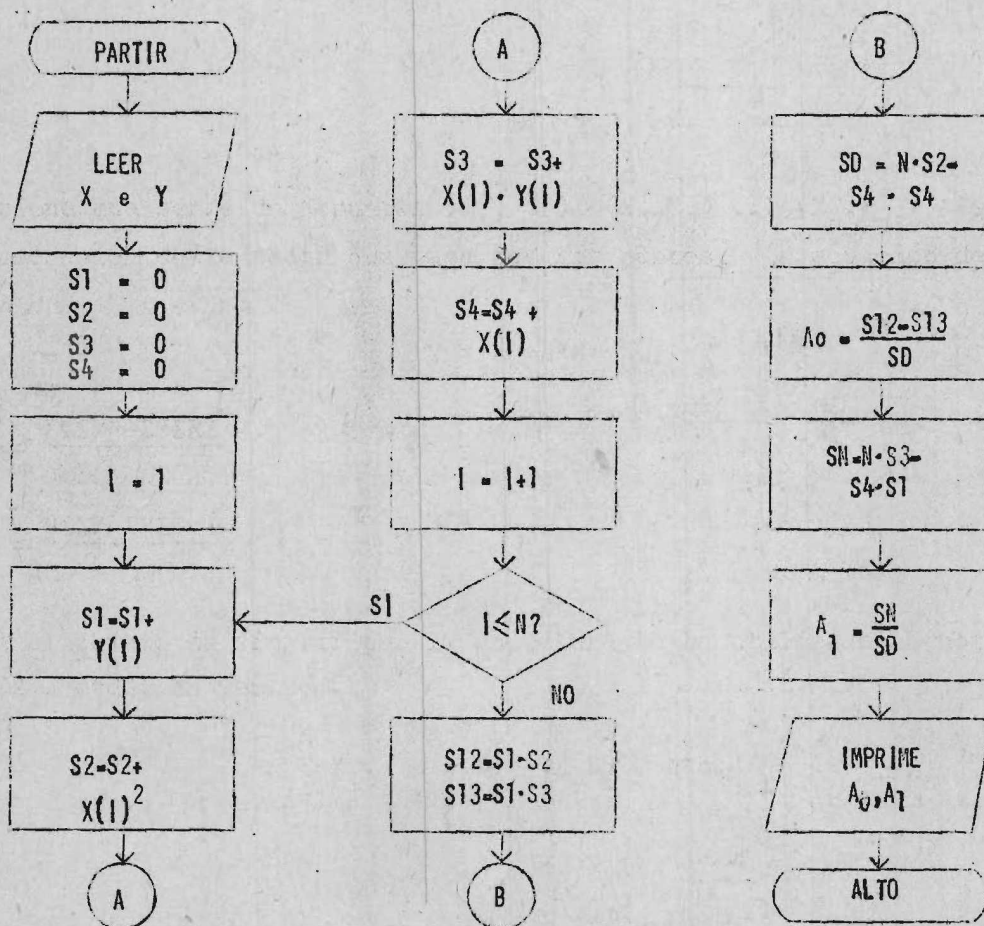
donde:

$$A_0 = \frac{\sum Y \sum X^2 - \sum X \sum Y}{N \sum X^2 - (\sum X)^2}$$

$$A_1 = \frac{N \sum X Y - \sum X \sum Y}{N \sum X^2 - (\sum X)^2}$$

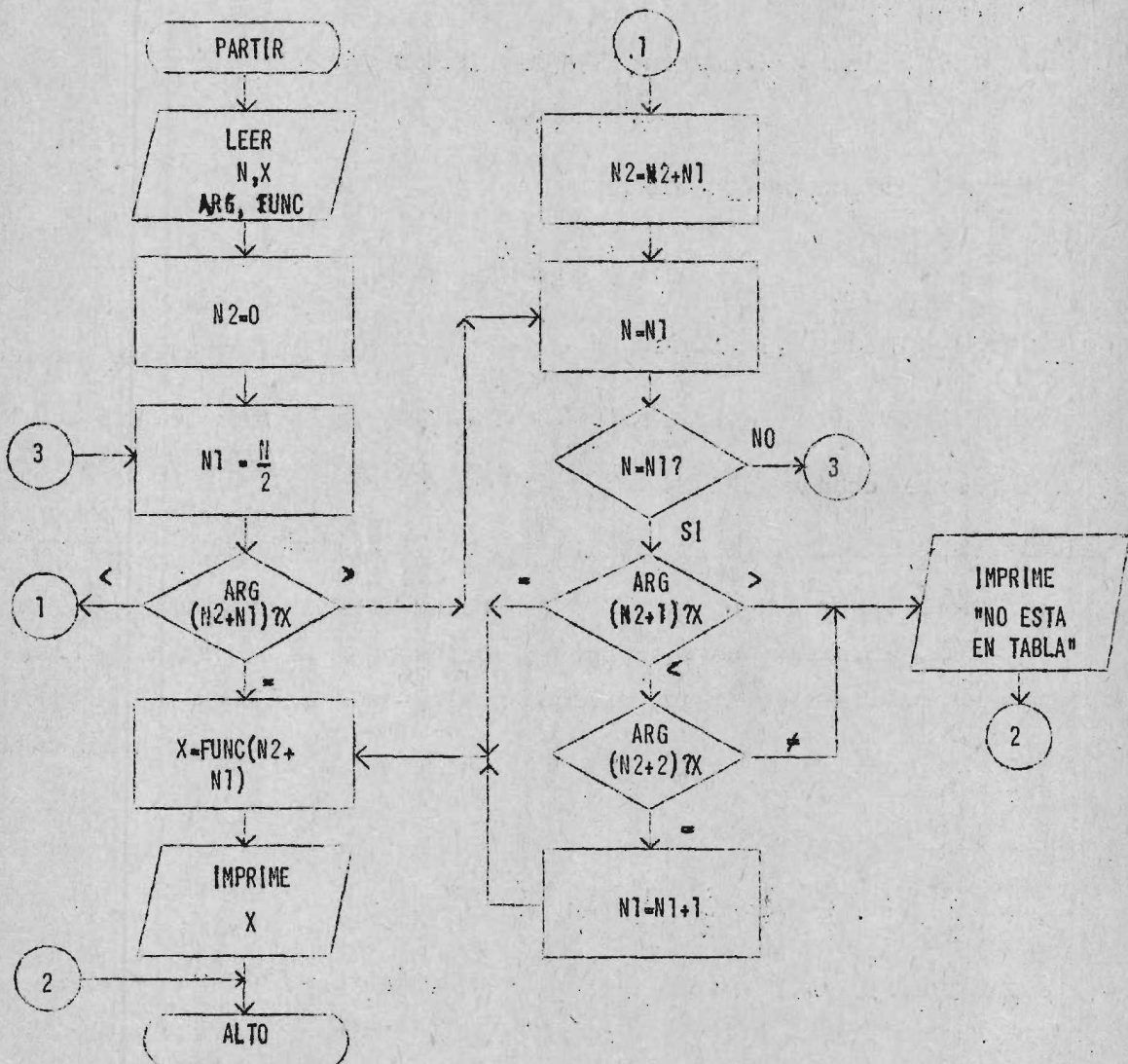
Con el objeto de simplificar la escritura se ha utilizado la notación $\sum X, \sum X Y, \sum X^2$, etc., en vez de:

$$\sum_{i=1}^M X_i, \sum_{i=1}^M X_i Y_i, \sum_{i=1}^M X_i^2, \text{ etc.}$$



7. Se tienen dos arreglos, ARGUMENTOS Y FUNCIONES cada una con N elementos (N puede ser par o impar). Se pide buscar un dato X en el primer arreglo con la técnica siguiente:

- Se compara el dato con el elemento que está en $N/2$
- Si es menor, se compara con el que está en $N/4$ y así sucesivamente.
- Si es mayor, se compara con el que está en $\frac{3}{4} N$, etc.
- Si es igual, el dato X se reemplaza por el elemento correspondiente del segundo arreglo.



VII. PROBLEMAS PROPUESTOS

1. Hallar la suma de 500 términos de la progresión aritmética cuyo primer término es 5 con incremento 4, esto es, 5, 9, 13, 17, ...

Utilizar la fórmula

$$S_n = \frac{[2 \cdot U_1 + (n-1)d] n}{2}$$

donde: U_1 = primer término

n = número de términos

d = incremento

S_n = sumatoria

2. Calcular las raíces de una ecuación de segundo grado del tipo

$$AX^2 + BX + C = 0$$

3. Resolver el sistema de ecuaciones

$$AX + BY = C$$

$$DX + EY = F$$

4. Calcular e^x a base de la fórmula

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} \dots$$

detener el cálculo cuando el último término sea menor que 10^{-7} .

5. Calcular el valor de π de acuerdo a la fórmula indicada por un dato leído N

a) Si $N=1$

$$\text{con } \frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9}$$

b) Si $N=2$

$$\text{con } \frac{\pi^2}{6} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \dots$$

c) Si $N=3$

$$\text{con } \frac{\pi}{2} = \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdot \frac{8}{7} \cdot \frac{8}{9}$$

6. Si se tienen los dos primeros elementos de un arreglo $N(1)=0$ y $N(2)=1$. Calcular hasta el término $N(1000)$ a base de la fórmula

$$N(I) = N(I-2) + N(I-1)$$

(Serie de Fibonacci)

7. Se tienen dos arreglos A y B, cada uno con 50 elementos. Se pide calcular

$$D = \sqrt{\sum_{i=1}^{50} (A_i - B_i)^2}$$

8. Se tiene un arreglo A con 50 elementos. Se pide transformarlo de acuerdo con:

$$a_i = a_i \cdot i$$

9. Se tiene un arreglo X de 50 por 50 elementos. Se pide encontrar el mayor de los elementos.

10. Se tiene un arreglo Y de 20 por 20 elementos. Se pide transformarlo en un arreglo YY de 400 elementos.

11. Se tiene un arreglo Z de 40 por 40 elementos. Se pide calcular

$$S = \sum_{i=1}^{40} Z_{ii}$$

12. En el arreglo anterior, se pide intercambiar el renglón K con la columna L. K y L son datos que deben ser leídos.

BIBLIOGRAFIA

1. CELADE, Curso de Introducción al Procesamiento Electrónico de Datos (PED) para Cientistas Sociales, Santiago, Chile, 1974, 117 p.
2. Gleim, George A., Program Flowcharting, San Francisco, Rinehart, 1970, 71 p.
3. IBM System Products Division, Técnicas de diagramación, Nueva York, 1970, 62 p.
4. Keenan, Thomas A., Forsythe, Alexander I., Organick, Elliot I. y Stenbert, Warren, Lenguajes de diagramas de flujo, México, Limusa/Wiley, 1973, 588 p.
5. Packer, David W., "Effective Program Design", en Computer and Automation, julio, 1970, vol. 19, N° 7, p. 37.



CENTRO LATINOAMERICANO DE DEMOGRAFIA
CELADE: J.M. Infante 9. Casilla 91. Teléfono 257806
Santiago (Chile)

CELADE: Ciudad Universitaria Rodrigo Facio
Apartado Postal 5249
San José (Costa Rica)