Directorio de Profesores del Curso: "ESTRUCTURAS DE DATOS" del 15 de Abril al 13 de Mayo de 1 9 8 3 .

1. ING. DANIEL RIOS ZERTUCHE ORTUÑO
Director de Informática
Subsecretaría de Planeación del Desarrollo
S P P
Izazaga No. 38-11° Piso
Col. Centro
Cuauhtémoc
México,D.F.
521 98 98 y 585 60 54

2. ING. RAYMUNDO H. RANGEL GUTIERREZ
Profesor
Edificio de Ingeniería Mecánica y Eléctrica
Anexo de Ingeniería
1° Piso Cubículo 11
UNAM
México, D.F.
550 52 15 Ext. 3746

3. ING. JORGE I. EUAN AVILA
Profesor
Edificio de Ing. Méc. y Eléctrica
Anexo de Ing. 1° Piso Cubículo 11
U N A M
México,D.F.
550 52 15 Ext. 3746

4. ING. LUIS G. CORDERO BORBOA (COORDINADOR)
Jefe del Departamento de Computación
Edif. de Ing. Méc. y Eléc.
Anexo de Ing. 1° Piso Cubículo 11
UNAM
México 20, D.F.
550 52 15 Ext.3746

5. Ing. Alejandro Jiménez García
Jefe del Centro de Cálculo
Facultad de Ingeniería
U N A M
México, D.F.

6. M. en C. Ricardo Ciria Merce
Subdirector de Diseño y Desarrollo
de Nuevos Proyectos
Coordinación de la Administración Escolar
Edificio IIMAS
p. B . Cubículo F
U N A M
México,D.F.

7. Ing. Rolando Samuel Carrera Sánchez

ESTRUCTURA DE DATOS

EJEMPLOS

I.- MINIEDITOR DE TEXTO
II.- INDEXADO KWIC
III.- DISPERSION EXTENSIVA
IV.- DISPERSION PERFECTA

ING. RAYMUNDO HUGO GUTIERREZ
MAYO, 1983

I MINIEDITOR DE TEXTO

FUNCIONES PRIMITIVAS

1.- Crear una cadena

CADENA ← 'ABCDEFG'

2.- Concatenar 2 cadenas para formar una sola cadena

CADENA ← ALFA OMEGA

3.- Buscar una replica de una cadena en otra cadena

i ← INDICE (OBJETO, PATRON)

4.- Extraer una subcadena de otra cadena

SCADENA ← SUB(OBJETO, i, j)

OBJETO. Cadena de la que se extrae una subcadena

i. Posicion del cursor que señala al primer caracter de la subcadena que se extrae.

j. Longitud (nro de caracteres) de la subcadena que se extrae

k. Longitud (nro de caracteres) de OBJETO

Si $j \leq 0$ regresa la cadena nula

Si $i \leq 0$ regresa la cadena nula

Si $i > k$ regresa la cadena nula

Si $i+j > k+1$ se asume que $j = k-i+1$

Si j no se da se asume que $j = k - i + 1$

5.- Reemplaza de una cadena por otra

SUB(OBJETO, i, j) ← X

6 - Longitud (nro de caracteres) de una cadena

i ← LONG(OBJETO)

FUNCIONES BASICAS y

LON, CASA, BARRER, HALLAR

Cada una de las funciones basicas tiene los siguientes argumentos:

OBJETO - Cadena en la que se busca la cadena PATRON

PATRON - Cadena que se busca en OBJETO

CURSOR - La posicion del primer caracter de la replica de PATRON que esta en OBJETO

COPIA - Variable a la que se asigna la replica de PATRON que esta en OBJETO

REEM - Cadena que reemplaza en OBJETO a la replica de PATRON, si BANREEM es verdad

BANREEM - Bandera que indica si o no se reemplaza la replica de PATRON que esta en OBJETO. Si BANREEM es verdad se hace el reemplazo, si es falso, no se hace el reemplazo.

Algoritmo LON. Dados los argumentos previamente descritos, LON regresa el valor verdad si hay NUM caracteres en OBJETO. La posicion de CURSOR es la del primer caracter de los NUM caracteres. Si hay NUM caracteres se asignan a COPIA. Si BANREEM es verdad, los NUM caracteres se reemplazan por REEM.

```
1.- {Verificar para NUM caracteres}
    Si (CURSOR + NUM > LONG(OBJETO) + 1) luego
        LON ← falso
        Salida
    fin
2.- {Asignar replica a copia y reemplazar si esta especificado}
    LON ← verdad
    COPIA ← SUB(OBJETO, CURSOR, NUM)
    Si (BANREEM) luego
        SUB(OBJETO, CURSOR, NUM) ← REEM
        CURSOR ← CURSOR + LONG(REEM)
        Salida
    fin
3.- {No hay reemplazo}
    CURSOR ← CURSOR + NUM
    Salida
```

LON (OBJETO, NUM, CURSOR, COPIA, REEM, BANREEM)

Ejemplo 1

OBJETO = 'SER O NO SER'

NUM = 2

CURSOR = 7

COPIA =

REEM = 'NUNCA'

BANRREM = verdad

Ejemplo 2

OBJETO = 'SER O NO SER'

NUM = 2

CURSOR = 14

COPIA =

REEM = 'NUNCA'

BANREEM = false

Ejemplo 3

OBJETO = 'SER O NO SER'

NUM = 2

CURSOR = 5

COPIA =

REEM = 'A B'

BANREEM = false

Ejemplo 4

OBJETO = 'SER O NO SER'

NUM = 6

CURSOR = 1

COPIA =

REEM = ''

BANREEM = verdad

Algoritmo CASA: Dados los 6 argumentos previamente descritos. CASA regresa el valor verdad si PATRON esta en OBJETO. La posicion de CURSOR es la del primer caracter de la replica de PATRON. Si PATRON esta en OBJETO, se asigna a COPIA el PATRON y si BANREEM es verdad la replica que esta en objeto se reemplaza por REEM.

```
1.- { Verificar si PATRON se encuentra dentro de los
      limites de OBJETO }
    Si (CURSOR + LONG(PATRON) > LONG(OBJETO) + 1) luego
        CASA ← falso
        Salida
    fin
2.- { Verificar si hay una replica de PATRON en OBJETO }
    Si (SUB(OBJETO, CURSOR, LONG(PATRON)) ≠ PATRON) luego
        CASA ← falso
        Salida
    fin
3.- { Reemplazar si se especifica }
    COPIA ← PATRON
    CASA ← verdad
    Si (BANREEM) luego
        SUB(OBJETO, CURSOR, LONG(PATRON)) ← REEM
        CURSOR ← CURSOR + LONG(REEM)
        Salida
    fin
4.- { No hay reemplazo }
    CURSOR ← CURSOR + LONG(PATRON)
    Salida
```

CASA(OBJETO, PATRON, CURSOR, COPIA, REEM, BANREEM)

Ejemplo 1

OBJETO = 'JUAN ESTUDIA INGENIERIA'
PATRON = 'JUAN'
CURSOR = 1
COPIA =
REEM = 'PEPE'
BANREEM = verdad.

Ejemplo 2

OBJETO = 'JUAN ESTUDIA INGENIERIA'.
PATRON = 'PEDRO'
CURSOR = 3
COPIA =
REEM = 'XAB'
BANREEM = falso

Ejemplo 3

OBJETO = 'JUAN ESTUDIA INGENIERIA'
PATRON = ''
CURSOR = 9
COPIA =
REEM = 'Y TRABAJA'
BANREEM = verdad

Algoritmo BARRER. Dados los 6 argumentos previamente descritas, BARRER regresa el valor verdad si la posicion, dada por CURSOR, de los caracteres en OBJETO se corresponden con los caracteres en PATRON. Si hay tal correspondencia, se asignan a COPIA los caracteres de OBJETO que se correspondan con los de PATRON. La comparacion finaliza cuando un caracter que no esta en PATRON se encuentra en OBJETO, o cuando el ultimo caracter en OBJETO se alcanza. La secuencia de caracteres comparados con exito se reemplaza por REEM si BANREEM es verdad.

```
1.- {Verificar si PATRON se encuentra dentro de los
     limites de OBJETO}
     Si (CURSOR > LONG(OBJETO)) luego
          BARRER ← falso
          Salida
     fin
2.- {Inicializar el indice a OBJETO con CURSOR}
     i ← CURSOR
3.- {Verificar si caracter i esta en PATRON}
     Entanto (i ≤ LONG(OBJETO) e INDICE(PATRON, SUB(OBJETO, i, 1)
                ≠ 0) repetir
          i ← i + 1
     fin
4.- {No se encontro caracteres correspondientes en PATRON}
     Si (i = CURSOR) luego
          BARRER ← falso
          Salida
     fin
5.- {Reemplazar si se especifica}
     BARRER ← Verdad
     COPIA ← SUB(OBJETO, CURSOR, i - CURSOR)
     Si (BANREEM) luego
          SUB(OBJETO, CURSOR, i - CURSOR) ← REEM
          CURSOR ← CURSOR + LONG(REEM)
          Salida
     fin
6.- {No hay reemplazo}  CURSOR ← i , Salida
```

BARRER(OBJETO, PATRON, CURSOR, COPIA, REEM, BANREEM)

Ejemplo 1

OBJETO = 'EL CAMINO.... Y CAMINO.... Y CAMINO'
PATRON = ','
CURSOR = 10
COPIA =
REEM = ','
BANREEM = verdad

Ejemplo 2

OBJETO = 'EL CAMINO.... Y CAMINO.... Y CAMINO'
PATRON = 'BI'
CURSOR = 1
COPIA =
REEM = ','
BANREEM = falso

Ejemplo 3

OBJETO = 'EL CAMINO.... Y CAMINO.... Y CAMINO'
PATRON = 'AMCONI'
CURSOR = 4
COPIA =
REEM = 'AXBD'
BANREEM = falso

Ejemplo 4

OBJETO = 'EL CAMINO.... Y CAMINO.... Y CAMINO'
PATRON = 'AZRCMIBNLOE.'
CURSOR = 1
COPIA =
REEM = ''
BANREEM = verdad

Algoritmo HALLAR. Dados los 6 argumentos previamente descritos, HALLAR regresa el valor verdad si PATRON se encuentra en cualquier lugar de OBJETO desde la posicion que tenga CURSOR en OBJETO hasta el extremo final de OBJETO. Si se encuentra una replica de PATRON en OBJETO, se asigna a COPIA la secuencia de caracteres que se encuentran entre la posicion de CURSOR y los caracteres que estan a la izquierda del primer caracter de la replica de PATRON en OBJETO. Si la replica de PATRON se encuentra comenzando con la posicion de CURSOR, a COPIA se asigna la cadena vacia. Si SANREEM es verdad, todos los caracteres comenzando desde la posicion de CURSOR hasta el caracter mas a la derecha de la reeplica se reemplaza por REEM

```
1.- { Verificar si PATRON se encuentra entre los limites
      de OBJETO }
    Si (CURSOR > LONG(OBJETO)) luego
        HALLAR ← falso
        Salida
    fin
2.- { Busqueda de la replica }
    I ← INDICE (SUB (OBJETO, CURSOR); PATRON)
    Si (i = 0) luego
        HALLAR ← falso
        Salida
    fin
3.- { Reemplazar }
    HALLAR ← verdad
    COPIA ← SUB(OBJETO, CURSOR, i - 1)
    Si ( SANREEM) luego
        SUB(OBJETO, CURSOR, LONG(PATRON) + i - 1) ← REEM
        CURSOR ← CURSOR + LONG(REEM)
        Salida
    fin
4.- { No hay reemplazo }
    CURSOR ← CURSOR + i + LONG (PATRON) - 1
    Salida.
```

HALLAR(OBJETO, PATRON, CURSOR, COPIA, REEM, BANREEM)

Ejemplo 1

OBJETO = 'LA ENCONTRÉ EL VERANO ANTES QUE YO VOLVIERA A LA ESCUELA'
PATRON = 'LA'
CURSOR = 3
COPIA =
REEM = 11
BANREEM = verdad

Ejemplo 2

OBJETO = 'LA ENCONTRÉ EL VERANO ANTES QUE YO VOLVIERA A LA ESCUELA'
PATRON = 'VOLVIO'
CURSOR = 1
COPIA =
REEM = 'XAX'
BANREEM = falso

Ejemplo 3

OBJETO = 'LA ENCONTRÉ EL VERANO ANTESQUE YO VOLVIERA A LA ESCUELA'
PATRON = 'LA'
CURSOR = 1
COPIA =
REEM = 'EL LA'
BANREEM = verdad

ANONIMA

Algoritmo $AGREGAR. Dado ENTRADA la primera linea de texto, el resto del texto de entrada se agrega al al cuerpo del texto cuanto no se encuentre el siguiente comando o indicador de fin de sesion. El indice CONLIN (contador de linea) es la posicion en el vector linea de caracteres en el que la entrada presente se almacena. La funcion COMANDOF verifica los comandos de formato antes de almacenar ENTRADA. COMANDOF se discute posteriormente. Por ahora asuma que COMANDOF es una funcion ficticia (devuelve el parametro ENTRADA sin alterar). L y C son variables intermedias y CONVACARA es una funcion que convierte un argumento numerico a una cadena de caracteres.

1 - { Repetir hasta que el comando $AGREGAR no tenga mas efecto }

Repetir de los pasos 2 a 5 cuanto no sea fin de sesion

2. - { Asignar contador de linea }
SUB(LINEA[CONLIN], 1, 5) ← '00000'
C ← CONVACARA(CONLIN * 10)
L ← LONG(C),
SUB(LINEA[CONLIN], 6-L, L) ← C

3. - { Almacenar ENTRADA }
SUB(LINEA[CONLIN], 11) ← COMANDOF(ENTRADA)

4. - { Incrementar contador de linea }
CONLIN ← CONLIN + 1

5. - { Leer nueva ENTRADA y verificar para un comando }
Leer ENTRADA
Si (SUB(ENTRADA, 1, 2) = '$$') Luego
Salida
fin

6. - { Fin de sesion }
Salida

$AGREGAR

SUPONIENDO QUE EL TEXTO ESTA EN TARJETAS PERFORADAS. EL TEXTO SE ASIGNA A UN VECTOR CUYOS ELEMENTOS SON CADENAS, EL TEXTO SE ALMACENA DE LAS POSICIONES 11 A LA 91 DE CADA ELEMENTO DEL VECTOR. UN ELEMENTO DE CADA VECTOR CORRESPONDE A UNA LINEA DE ENTRADA. LAS POSICIONES DE LA 1 A LA 5 DE CADA ELEMENTO CORRESPONDE A UN NUMERO DE SECUENCIA DE LINEA. PARA CADA TEXTO LOS NUMEROS DE SECUENCIA COMIENZAN CON 00010 Y SE INCREMENTAN DE 10 EN 10

Algoritmo $$LISTAR. Dado el vector LINEA y los parametros INICIOLINEA y FINLINEA los elementos apropiados de linea son impresos en los indices y se usa en la salida de los elementos de LINEA.

1.- { Salida del texto especificado }

Desde j:=INICIOLINEA hasta FINLINEA repetir
Si (SUB(LINEA[j], 11) ≠ 11) luego
imprimir SUB(LINEA[j], 1, 5) o 'bbbbb' o
SUB(LINEA[j], 11)

fin

2.- { fin }

Salida

$$LISTAR/NUMERO DE LINEA INICIAL/NÚMERO DE LINEA FINAL

$$LISTAR/00010/00090

00010 ASUMIENDO QUE EL TEXTO ESTA EN TARJETAS PERFORADAS. EL TEXTO
00020 SE ASIGNA A UN VECTOR CUYOS ELEMENTOS SON CADENAS. EL TEXTO
00030 SE ALMACENA DE LAS POSICIONES 11 A LA 91 DE CADA
00040 ELEMENTO DEL VECTOR. UN ELEMENTO DE CADA VECTOR
00050 CORRESPONDE A UNA LINEA DE ENTRADA. LAS POSICIONES
00060 DE LA 1 A LA 5 DE CADA ELEMENTO CORRESPONDE A
00070 UN NUMERO DE SECUENCIA DE LINEA. PARA CADA TEXTO
00080 LOS NUMEROS DE SECUENCIA COMIENZAN CON 00010
00090 Y SE INCREMENTAN DE 10 EN 10.

$$LISTAR/00030/00040

00030 SE ALMACENA DE LAS POSICIONES 11 A LA 91. DE CADA
00040 ELEMENTO DEL VECTOR. UN ELEMENTO DE CADA VECTOR

$$LISTAR/00080/00090/

00080 LOS NUMEROS DE SECUENCIA COMIENZAN CON 00010
00090 Y SE INCREMENTAN DE 10 EN 10

ALGORITMA PARA CAMBIAR UN TEXTO

Algoritmo $$CAMBIA Dados los 4 parametros INICIOLINEA FINLINEA, PATRON y REEM, las lineas designadas por la secuencia de lineas de INICIOLINEA a FINLINEA en el vector LINEA se cambian al sustituir el texto PATRON por el texto REEM. La variable j es un indice para el vector LINEA, IND se usa como una variable temporal y CAMBIOBAN indica si al menos una ocurrencia de cambio tuvo lugar

1.- {Iniciar iteracion}
    CAMBIOBAN ← falso
    Para los pasos 2 y 3
    Desde j ← INICIOLINEA hata FINLINEA repetir
2.- {Localizar texto a cambiar}
    IND ← INDEX(LINEA[j], PATRON)
3.- {Realizar sustitucion si es posible}
    Si (IND ≠ 0) luego
        SUB(LINEA[j], IND, LONG(PATRON)) ← REEM
        CAMBIOBAN ← verdad
    fin
4.- {fin}
    Si (¬CAMBIOBAN) luego
        imprimir 'TEXTO HA CAMBIARSE NO SE LOCALIZO
    fin

$$CAMBIA/NUMERO DE LINEA INICIAL/NUMERO DE LINEA FINAL
<TEXTO QUE SE SUSTITUYE>/<TEXTO QUE SE INSERTA>

$$ CAMBIA /00040/00040/VECTOR/ARREGLO UNIDIMENSIONAL

$$ CAMBIA/00060/00060/DE CADA ELEMENTO//

$$SUPRIMIR/NUMERO DE LINEA INICIAL/NRO DE LINEA FINAL/

$$SUPRIMIR/00050/00050/

$$ SUPRIMIR/00020/00050/

$$ SUPRIMIR/00060/*/

Algoritmo $$SUPRIMIR. Dados los 2 parametros INICIOLINEA y FINLINEA el conjunto de lineas entre (e incluyendo las) INICIOLINEA y FINLINEA se suprimen

1.- {-Suprimir lineas}

Desde i ← INICIOLINEA hasta FINLINEA

LINEA[i] ← ''

fin

2.- {fmi}

Salida.

CODIGOS DE FORMATO

COMANDO DE MARGENES

@@MARGEN <conjunto de margenes> @|T

Ejemplo

```
@@MARGEN/15/30/45/@KMS/KMS//LITRO/COSTO/
200/19.5/$5.25/250/21.0/$6.85/
195/16.4/$5.20/
$$LISTAR/00010/*/
$$IMPRIMIR/00010/*/

00010   @@15/30/45/@KMS/KMS//LITRO/COSTO/
00020   200/19.5/$5.25/250/21.0/$6.85/
00030   195/16.4/$5.20/
```

KMS	KMS/LITRO	COSTO
200	19.5	$5.25
250	21.0	$6.85
195	16.4	$5.20

$$AGREGAR

@@MARGEN/15/26/93/KMS/KMS//LITRO/COSTO/
@@MARGEN/15/29/93/@200/19.95/$5.25/%
250/21.0/$6.85/195/16.9/$5.20/
$$IMPRIMIR/00010/*/

KMS	KMS/LITRO	COSTO
200	19.5	$ 5.25
250	21.0	$ 6.85
195	16.9	$ 5.20

$$AGREGAR
@@MARGEN/5/ESTE ES EL COMIENZO DE UN PARRAFO
Y ES USADO PARA PROPOSITOS ILUSTRATIVOS
$$IMPRIMIR/00010/*/

ESTE ES EL COMIENZO DE UN PARRAFO
Y ES USADO PARA PROPOSITOS ILUSTRATIVOS

C centrar
S subrayado
J justificado a la izquierda

COMANDO PARA TITULO

%%TITULO/C,Ó,I/S Ó N/< TEXTO >

$$AGREGAR
%%TITULO/C S/INTRODUCCION A LAS ESTRUCTURAS DE DATOS
%%TITULO/C N/POR
%%TITULO/C N/J. P. TREMBLAY
%%TITULO/C N/P. G. SORENSON
%%TITULO/I S/PUBLICADO POR
%%TITULO/I N/MC GRAW-HILL
$$IMPRIMIR/00010/X/

INTRODUCCION A LAS ESTRUCTURAS DE DATOS

POR
J. P. TREMBLAY
P. G. SORENSON

PUBLICADO POR

MCGRAW-HILL

COMANDO_DE_SALTO

#/#SALTO/<NRO DE LINEAS>/P/

$$AGREGAR
#/#SALTO/P/
%/TITULO/CN/CAPITULO 2
#/#$ALTO/1/
%/%/TITULO/CN/MANIPULACION DE CADENAS
#/#SALTO/1/
@@MARGEN/5/EN EL CAPITULO PREVIO SE INTRODUJO EL CARACTER

CAPITULO 2

MANIPULACION DE CADENAS

EN EL CAPITULO PREVIO SE INTRODUJO EL CARACTER

COMANDO DE JUSTIFICACION

&&JUSTIFICAR/[?]<posicion de margen derecho>/

Algoritmo JUSTIFICACION. Dada la cadena IMPLINEA que contiene un texto con caracteres no blancos al inicio y al final y de longitud mayor que MARGEND, IMPLINEA justifica a la derecha y cualquier exceso de texto se regresa. BLANCOS es una variable que tiene el numero de blancos que han de insertarse y CAMPOB es una cadena de blancos igual en longitud a la longitud de el campo de blancos que separa a las palabras. Inicialmente el tamaño de este campo es uno.

1.-{Verificar si el texto es inmediatamente justificable a la derecha}

Si (SUB(IMPLINEA, MARGEND, 1) ≠ 'b' y SUB(IMPLINEA, MARGEND+1, 1) = 'b')

imprimir SUB(IMPLINEA, 1, MARCEND)

fin

JUSTIFICACION ← SUB(IMPLINEA, MARGEND+1)

Salida

2.-{Verificar si la posicion de MARGEND es un no blanco}

j ← MARGEND - 1

Si (SUB(IMPLINEA, MARGEND, 1) ≠ 'b') luego

Entanto (SUB(IMPLINEA, j, 1) ≠ 'b') repetir

j ← j-1

fin

fin

3.-{Buscar siguiente caracter no blanco}

j ← j+1

Entanto (SUB(IMPLINEA, j, 1) = 'b') repetir

j ← j-1

fin

4.-{Inicie iteracion para agregar blancos}

BLANCOS ← MARGEND - j

CAMPOB ← 'b'

Repetir para el paso 5

Desde K ← 1 hasta BLANCOS repetir

5.- {Sucesivamente agregar al campo blanco que separa a las palabras}

Entanto (¬CASAR(IMPLINEA, CAMPOB, j, " ", CAMPOB, 'b', verdad)

j ← j - 1

Si (j = 0) luego

j ← MARGEND - BLANCOS + K - 1

CAMPOB ← CAMPOB + 'b'

fin fin

j ← j - LONG(CAMPOB) - 2

6.- {Salida de texto justificado}

IMPRIMIR SUB(IMPLINEA, 1, MARGEND)

JUSTIFICACION ← SUB(IMPLINEA, MARGEND + 1)

Salida.

EL LIBRO FUE ESCRITO POR W. MC FINDLING EL DISCUTE BASES RELACIONALES...

Algoritmo #IMPRIMIR. Dado el texto almacenado en LINEA y los parametros de numeros de linea: INICIOLINEA y FINLINEA, el texto entre, e inclusive, LINEA [INICIOLINEA] y LINEA [FINLINEA] se imprime de acuerdo al formato dictado por los codigos incluidos en el texto. SEMARGEN y SEJUSTI son variables logicas usadas para indicar cuando los controles de justificacion y margen se encuentran en una busqueda comenzando en INICIOLINEA y decrementando hasta llegar a la primera linea de entrada. BANMARGEN indica si el control de margen actual es global, local o se deja sin efecto. BANMARGEN puede tomar los valores G, L o N respectivamente. NOMARGEN tiene el numero de margenes actuales. Cada valor de un margen se almacena en el vector MARGEN. JUSTID es una variable logica que cuando es verdad indica que el texto que sigue se justifica a la derecha y cuando es falso indica que el texto que sigue no se justifica a la derecha. MARGEND tiene el valor del margen derecho.

```
1.-{Inicializacion de busqueda de josthficacion y margen previos}
   SEMARGEN ← SEJUSTI ← falso
   CURSOR ← 13
   Para los pasos 2 y 4
   Desde i ← INICIOLINEA. hasta 1 repetir
2.-{Verificar si codigos para margen y josthficacion se localizaron}
   Si (SEMARGEN y SEJUSTI) luego
        ir al paso 5
   fin
3.-{Verificar si LINEA[i] contiene un codigo de margen}
   Si (¬SEMARGEN y SUB(LINEA[i],11,2) = '@@') luego
        SEMARGEN ← verdad
        FICTICIO ← BARRER(LINEA[i],'0123456789/',CURSOR,LISTAM,11,falso)
        Si (SUB(LINEA[i],CURSOR,1)='@') luego
             BANMARGEN ← '6'
             ASIMARGEN(LISTAM)
        obien
             BANMARGEN ← 'L'
        fin
4.-{Examine LINEA[i] para un codigo de josthficacion}
   Si (¬SEJUSTI y SUB(LINE[i],11,2)='kk') luego
        SEJUSTI ← verdad
        Si (SUB(LINEA[i],13,1)='7') luego
             JUSTID ← falso
             C ← 14
        obien
             JUSTID ← verdad
             C ← 13
        fin
        MARGEND ← SUB(LINEA[i],C,INDICE(SUB(LINEA[i],C),'/')-1)
   fin
5.-{Se inicializa fase de impresion}
   i ← INICIOLINEA - 1
   IMPLINEA ← ''
6.-{Comienza fase de impresion}
   i ← i + 1
   Si (i > FINLINEA) luego
        imprimir IMPLINEA
        salida
   fin
   IMPLINEA ← IMPLINEA ⊙ SUB(LINEA[i],11)
```

```
7.-{Verificar para codigos de salto}
    Si ( SUB(LINEA[i],11,2 ) = '##') luego
        Imprimir IMPLINEA
        Si ( SUB(LINEA[i],13,1) ='p' ) luego
            saltar a nva pagina.
        obien
            saltar SUB(LINE[i],13,INDICE(SUB(LINEA[i],13),'/')-1)
                Lineas
        fin
        IMPLINEA ← SUB(LINEA[i],14+INDICE(SUB(LINEA[i],14),'/'))
8.-{Verificar para titulos de codigo}
    Si ( SUB(LINEA[i],11,2 ) ='%%') Luego
        imprimir IMPLINEA
        Si (SUB(LINEA[i],13,1) = 'c' ) luego
            CENTRAR(LINEA[i],MARGEND)
        obien
            imprimir SUB(LINEA[i],16)
            IMPLINEA ← ''
            Si ( SUB(LINEA[i],14,1) = 'U' ) luego
                imprimir subrayado
            fin
        fin
9.-{Verificar para codigo de margen}
    Si ( SUB(LINEA[i],11,2) ='@@') luego
        imprimir IMPLINEA
        CURSOR ← 13
        FICTICIO ←BARRER(LINEA[i],'0123456789/',CURSOR,LISTAM,'',falso)
        ASIMARGEN(LISTAM)
        Si ( SUB(LINEA[i],CURSOR,1) = '@' ) luego
            BANMARGEN ←'@'
        obien
            BANMARGEN ← 'L'
        fin
        IMPLINEA ← SUB(LINEA[i],CURSOR+1)
        Si ( IMPLINEA = '' ) luego
            ir al paso 6
        fin
```

```
10.-{Verificar para codigo de justificacion}
   Si (SUB(LINEA[i],11,2)='b&b') luego
      Si (SUB(LINEA[i],13,1)='7') luego
         JUSTID ← falso
         MARGEND ← SUB(LINEA[i],14,INDICE(SUB(LINEA[i],14,'/')-1)
      obien
         JUSTID ← verdad
         MARGEND ← SUB(LINEA[i],13,INDICE(SUB(LINEA[i],13,'/')-1)
      fin
11.-{Si es el caso manejar margenes}
      Si (BANMARGEN='L' o BANMARGEN='G') luego
         Si (NOMARGENES > 1) luego
            IMPMARGEN(IMPLINEA)
            ir al paso 6
         obien
            IMPLINEA ← DUPL('b',MARGEN[i])⊙IMPLINEA
         fin
12.-{Repeticion para impresion de linea}
      Repetir pasos 13 y 14
      ENTANTO(LONG(IMPLINEA) ≥ MARGEND) repetir
13.-{Manejar justificacion derecha}
      Si (JUSTID) luego
         IMPLINEA ← JUSTIFICACION(IMPLINEA,MARGEND)
      obien
         IMPLINEA ← NOJUSTIFICAR(IMPLINEA,MARGEND)
      fin
14.-{Establecer margenes}
      FICTICIO ← BARRER(IMPLINEA,'b',1,11,11,verdad)
      Si (BANMARGEN='G') luego
         IMPLINEA ← DUPL('b',MARGEN[1])⊙IMPLINEA
      fin.
15.-{Actualice BANMARGEN}
      Si (BANMARGEN='L') luego
         BANMARGEN ← 'N'
      fin
      ir al paso 6
```

Algoritmo IMPMARGEN. Dado el parámetro IMPLINEA y el vector MARGEN, el texto en IMPLINEA se copia formateado LINEASAL y se imprime LINEASAL. NOMARGEN es el numero de margenes e i es un contador

```
1.- {Inicializacion}
    i ← 1
    LINEASAL ← ''
    CURSOR ← 1
2.- {Busqueda del separador /}
    Repetir pasos 3 a 5
    Entanto (FIND(IMPLINEA, '/', CURSOR, COPIA, falso) repetir
3.- {Buscar //}
    Si (SUB(IMPLINEA, CURSOR, 1) = '/') luego
        TEMP ← TEMP O COPIA O '/'
        CURSOR ← CURSOR + 1
        Ir al paso 2
    obien
        TEMP ← TEMP O COPIA
    fin
4.- {Colocar el valor de TEMP en la posicion correcta en LINEASAL}
    SUB(LINEASAL, MARGEN[i], LONG(TEMP)) ← TEMP
    TEMP ← ''
5.- {Actualizar i y verificar si es menor que
    i ← i + 1
    Si (i > NOMARGEN) luego
        IMPLINEA ← SUB(IMPLINEA, CURSOR)
        Imprimir LINEASAL
        i ← 1
        LINEASAL ← ''
    fin
6.- {fin}
    Imprimir LINEASAL
    Salida.
```

Algoritmo INTERPRETAC. Dada la cadena ENTRADA esta cadena se examina en busca de comandos. CONLIN es el indice de una LINEA asociado con la siguiente linea disponible de texto y HALLARLIN es un algoritmo que calcula INICIOLINEA y FINLINEA.

1. {Procesar entrada hasta el fin de sesion?
   Repetir de los pasos 2 a 9 hasta fin de sesion
2. {Obtener siguiente linea e imprimirla}
   Leer ENTRADA
   Imprimir ENTRADA
3. {¿$$ Comandos?}
   Si (SUB(ENTRADA,1,2) ≠ '$$') luego
      imprimir 'ENTRADA ILEGAL' e ir a paso 1
   fin
4. {Procesar comandos comenzando con $$AGREGAR}
   Si (SUB(ENTRADA,1,9) = '$$AGREGAR') luego
      llamar $$ADD e ir a paso 1
   fin
5. {¿$$LISTAR?}
   CURSOR ← 10
   Si (SUB(ENTRADA,1,8) = '$$LISTAR') luego
      HALLARLIN(ENTRADA,CURSOR)
      $$LISTAR(INICIOLINEA,FINLINEA)
      ir al paso 1
   fin
6. {¿$$ CAMBIA?}
   CURSOR ← 10
   Si (SUB(ENTRADA,1,8) = '$$CAMBIA') luego
      HALLARLIN(ENTRADA,CURSOR)
      $$CAMBIA(INICIOLINEA,FINLINEA)
      ir al paso 1
   fin

7.- {¿$$ SUPRIMIR?}
CURSOR ← 12
si (SUB(ENTRADA, 1, 10) = '$$SUPRIMIR') luego
HALLARLIM(ENTRADA, CURSOR)
$$SUPRIMIR(INICIOLINEA, FINLINEA)
ir al paso 1
fin

8.- {$$IMPRIMIR?}
CURSOR ← 12
si (SUB(ENTRADA, 1, 10) = '$$IMPRIMIR') luego
HALLARLIM(ENTRADA, CURSOR)
$$IMPRIMIR(INICIOLINEA, FINLINEA)
ir al paso 1
fin

9.- {Error en ENTRADA}
Imprimir 'COMANDO ILEGAL'
ir al paso 1

Algoritmo HALLARLIM. Dada la cadena ENTRADA y la posicion del cursor indicando el inicio del numero de linea para un comando particular, los valores para INICIOLINEA y FINLINEA se calculan

```
1.- {Aislar campos de parametros para INICIOLINEA y FINLINEA}
    Si (HALLAR(ENTRADA,'/',CURSOR,INICIOLINEA,'',falso)) luego
        Si (HALLAR(ENTRADA,'/',CURSOR,FINLINEA,'',falso)) luego
            imprimir 'ERROR - FINLINEA, OMISION DE PARAMETROS'
            Salida
        fin
    obien
        IMPRIMIR 'ERROR - OMISION DE PARAMETROS'
        Salida
    fin
2.- {Verifique para * y asigne ultima linea}
    Si (INICIOLINEA = '*') Luego
        INICIOLINEA ← CONLIN - 1
        Salida
    fin
    Si (FINLINEA = '*') luego
        FINLINEA ← CONLIN - 1
        Salida
    fin
    INICIOLINEA ← INICIOLINEA/10
    FINLINEA ← FINLINEA/10
    Salida.
```

Algoritmo COMANDOF. Dada la cadena ENTRADA se barre en busca de comandos de formato. Una vez hallado el comando de formato se interpreta y el código apropiado de formato se almacena.

1.- {Verifique para comandos de formato y sustituye palabras clave}

Si (MATCH(ENTRADA, '@@TAB/', 1, '', '@@', verdad) luego
ir al paso 2
fin
Si (MATCH(ENTRADA '%%TITULO/', 1, '', '%%', verdad) luego
ir al paso 2
fin
Si (MATCH(ENTRADA, '##SALTO/', 1, '', '##', verdad) luego
ir al paso 2
fin

FICTICIO ← MATCH(ENTRADA, '&&JUSTIFICAR/', 1, '', '&&', verdad)

2.- { Asignar valor de regreso y fin }

COMANDOF ← ENTRADA
Salida.

```
$$ADD (ABRGEAR)
&&JUSTIFY/70/ (JUSTFICAR)
@@TAB/40/@187 MAIN STREET (MARGEN)
WINNIPEG 1, MANITOBA
*DATE*
##SKIP/1/ (SALTO)
@@TAB/0/@ (MARGEN)
*X*
*ADDRESS*
*CITY*, *PROVINCE*

##SKIP/1/ (SALTO)
DEAR *Z*,
##SKIP/1/ (SALTO)
&&JUSTIFY/70/ (JUSTFICAR)
@@TAB/5/
THE BUSINESS WORLD IS RAPIDLY CHANGING AND OUR CORPORATION HAS
BEEN KEEPING PACE WITH THE NEW REQUIREMENTS FORCED UPON OFFICE
MACHINERY. WE ARE GIVING YOU, *Z*, AS A KEY FIGURE IN THE *CITY*
BUSINESS COMMUNITY, AN OPPORTUNITY TO BECOME FAMILIAR WITH THE
LATEST ADVANCEMENTS IN OUR EQUIPMENT. A REPRESENTATIVE OF OUR
CORPORATION IN *PROVINCE* WILL BE SEEING YOU WITHIN *N* WEEKS. HE
WILL TAKE SEVERAL MACHINES TO *CITY* WHICH ARE INDICATIVE OF A
WHOLE NEW LINE OF OFFICE MACHINES WE HAVE RECENTLY DEVELOPED.
@@TAB/5/ OUR SALES REPRESENTATIVE IS LOOKING FORWARD TO HIS
VISIT IN *CITY*. HE KNOWS THAT THE MACHINES HE SELLS COULD BECOME
AN INTEGRAL PART OF YOUR OFFICE ONLY A FEW DAYS AFTER INSTALLATION.
##SKIP/1/ (SALTO)
&&JUSTIFY/70/ (JUSTIFICAR)
@@TAB/40/@ (MARGEN)
SINCERELY,
##SKIP/1/ (SALTO)
ROGER SMITH, MANAGER
OFFICE DEVICES INCORPORATED
##SKIP/P/ (SALTO)
```

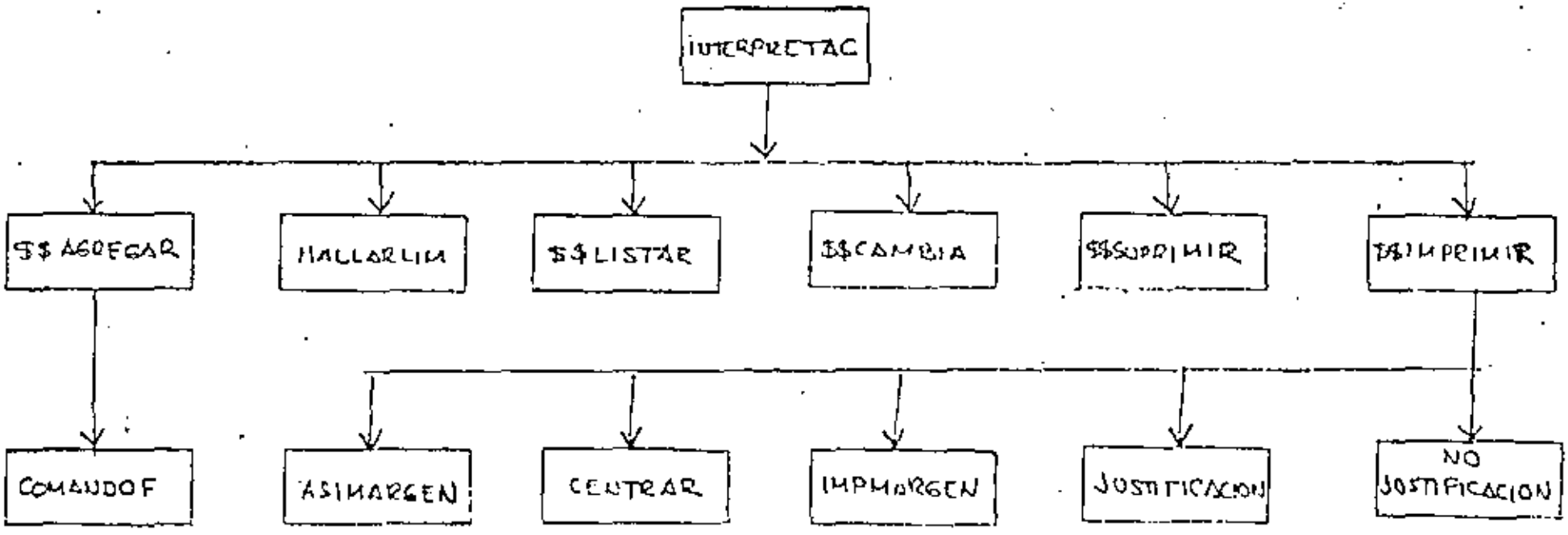
The general letter is introduced into memory and a copy of the letter is stored in an auxiliary file for later processing. Next the fields of text which are delineated by *'s are changed, based on the following specific information:

1. Date (e.g., August 17, 1975)
2. MR. (or MRS., etc.), initial, surname (e.g., Mr. A.L. Strider)
3. Street address (e.g., 2014 Centennial Drive)
4. City, Province(or State) (e.g., Thompson, Manitoba)
5. *N*, the number of weeks before salesman will visit (e.g., three)

An ETEXTE session for creating a personal letter proceeds as follows:

CHANGE = CAMBIA

```
$$CHANGE/00010/*/*DATE*/AUGUST 17, 1975/
$$CHANGE/00010/*/*ADDRESS*/2014 CENTENNIAL DRIVE/
$$CHANGE/00010/*/*CITY*/THOMPSON/
$$CHANGE/00010/*/*PROVINCE*/MANITOBA/
$$CHANGE/00010/*/*N*/THREE/
$$CHANGE/00010/*/*X*/MR. A.L. STRIDER/
$$CHANGE/00010/*/*Z*/MR. STRIDER/
$$PRINT/00010/*/
```


INTERPRETAC
$$ AGREGAR
HALLARLIN
$$LISTAR
$$CAMBIA
$$SUPRIMIR
$$IMPRIMIR
COMANDOF
ASIMARGEN
CENTRAR
IMPMARGEN
JUSTIFICACION
NO JUSTIFICACION

indexado KWIC (Key-Word-In-Context)

permite determinar el papel de una palabra rapidamentemente.
Las palabras claves se eligen de tal modo que tengan algun significado a la naturaleza del documento.

Ejemplo

'UNA INTRODUCCION A LA ESTRUCTURA DE DATOS CON APLICACIONES //'

VECTOR TITULO

TITULO[1] = 'UNA INTRODUCCION A LA ESTRUCTURA DE DATOS CON APLICACIONES //'

TITULO[2] = 'UNA INTRODUCCION A LA PROGRAMACION //'

TITULO[3] = 'PROGRAMACION PL/I CON APLICACIONES //'

TITULO[4] = 'UNA INTRODUCCION A SNOBOL4 //'

TITULO[5] = 'UNA INTRODUCCION A LA PROGRAMACION LISP //'

i	VECTOR PALCLAVE	TITULO#C
1	'APLICACIONES'	'1 3'
2	'DATOS'	'1'
3	'INTRODUCCION'	'1 2 4 5'
4	'LISP'	'5'
5	'PL/I'	'3'
6	'PROGRAMACION'	'2 3 5'
7	'SNOBOL 4'	'4'
8	'ESTRUCTURA'	'1'

Algoritmo FUERAKWIC. Dados los arreglos TITULO, PALCLAVE y TITULO#C, este algoritmo genera un indice KWIC ordenado lexicamente por palabras indice. LLAVEC es una variable intermedia usada para mantener la cadena de indices de TITULO tal como estan almacenados en TITULO#C. IND mantiene un indice particular del arreglo TITULO, LLAVEULTIMA es el nro de palabras clave almacenadas y T se usa en la formacion de un indice permutado.

1.- {Iterar}
    Repetir pasos 2 a 5
    Desde i ← 1 hasta LLAVEULTIMA repetir
2.- {Asignar a LLAVEC}
    LLAVEC ← TITULO#S[i] O 'b'
3.- {Repetir hasta que no haya indices TITULO en LLAVEC}
    Repetir pasos 4 a 5
    En tanto (LONG(LLAVEC))>1) repetir
4.- {Obtener siguiente indice TITULO
    IND ← SUB(LLAVEC, 1, INDICE(LLAVEC, 'b')-1)
    LLAVEC ← SUB(LLAVEC, INDICE(LLAVEC, 'b')+1)
5.- {Dar salida a T en formato KWIC}
    Si (HALLAR(T, PALCLAVE[i], CURSOR, COPIA, 11, true) luego
        T ← PALCLAVE[i] O SUB(T, CURSOR) O 'b' O COPIA
        imprimir T
    obien
        imprimir 'PALABRA CLAVE NO SE ENCONTRO
    fin;
6.- {Fin}
    Salida.

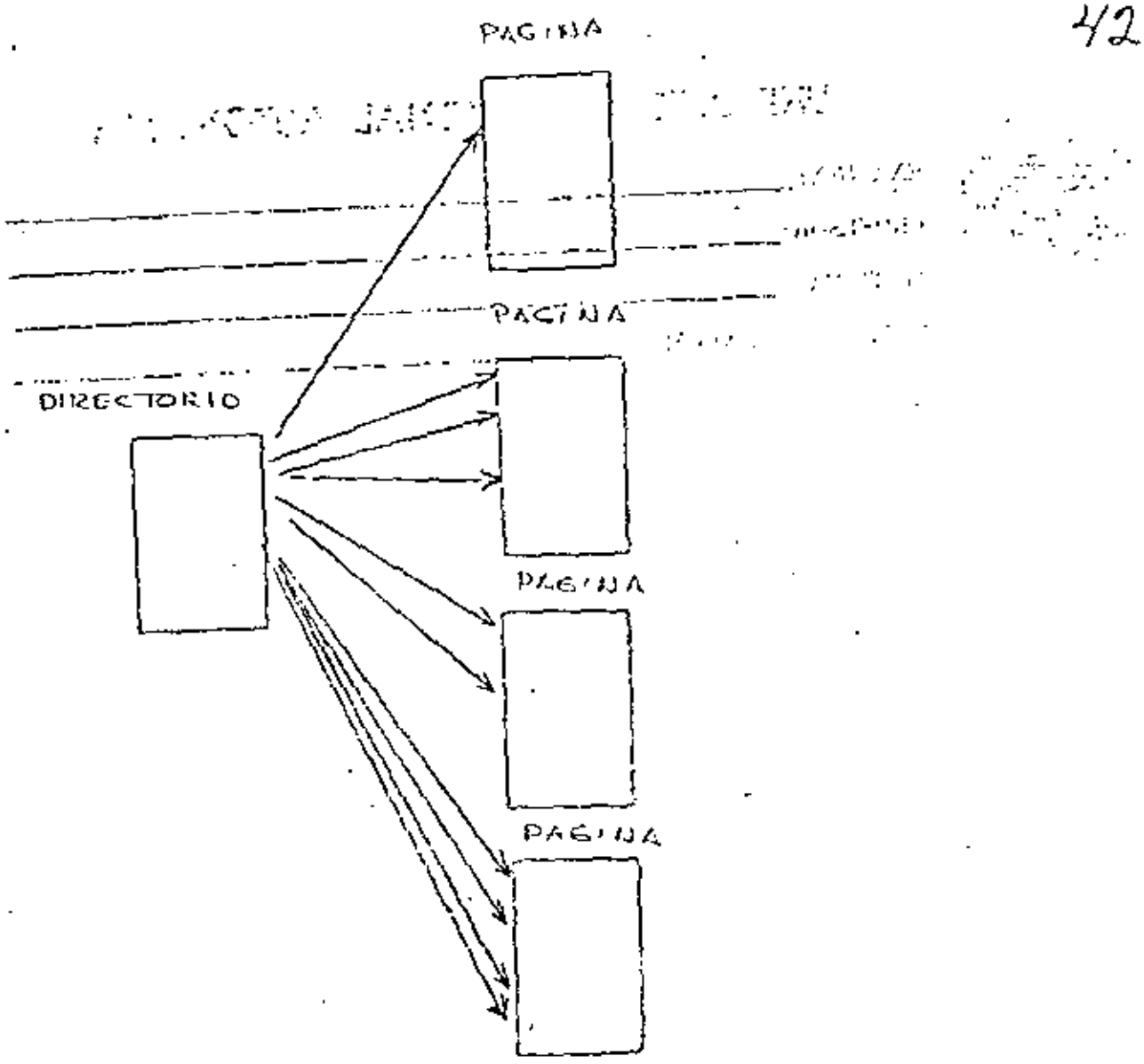
III DISPERSION EXTENSIVA (EXTENDIBLE HASHING)

1.- Es un metodo para accesar archivos que estan cambiando constantemente y experimentando frecuentes actualizaciones

2.- Esta diseñado para localizar cualquier registro de datos en no mas de 2 accesos al medio de almacen externo que contiene el archivo

3.- La tabla de indices puede expanderse y contraerse a medida que el archivo se expande y contrae

4.- La estructura de la dispersion extensiva esta compuesta de paginas y un directorio. Una pagina es una area de tamaño fijo que contiene registros de datos o apuntadores a otros registros. Un directorio contiene solo apuntadores a paginas.



EL algoritmo basico de busqueda es como sigue:

1.- Dada una llave, apliquele la funcion de dispersion.

2.- Use el resultado para localizar una entrada en el directorio

3.- Tome el apuntador correspondiente que esta en la entrada localizada en 2 para accesar la pagina

4.- Buscar la llave en la pagina. Si la llave no esta en la pagina, entonces no esta en el archivo.

INICIALMENTE

SUPONGAMOS QUE TENEMOS 16 REGISTROS Y QUE LA LLAVE DE CADA REGISTRO TIENE 16 CARACTERES DE LARGO

LA FUNCION DE DISPERSION ES COMO SIGUE

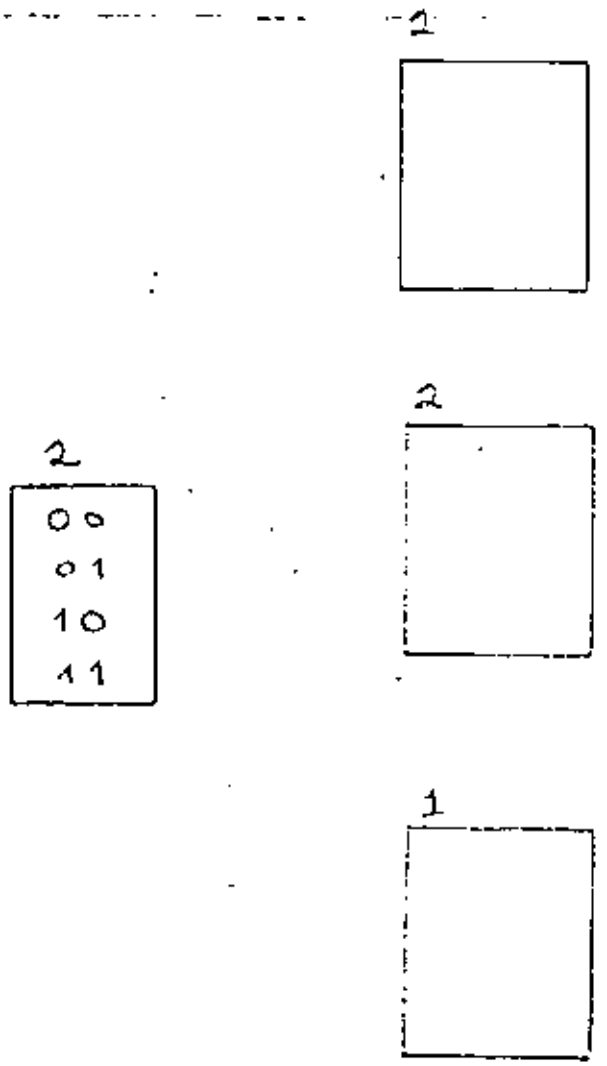
1.- SE DOBLA LA LLAVE

2.-

EL RANGO DE LA FUNCION DISPERSADA VA DE 0 A $2^{31} - 1$ (= 2 147 483 647.)

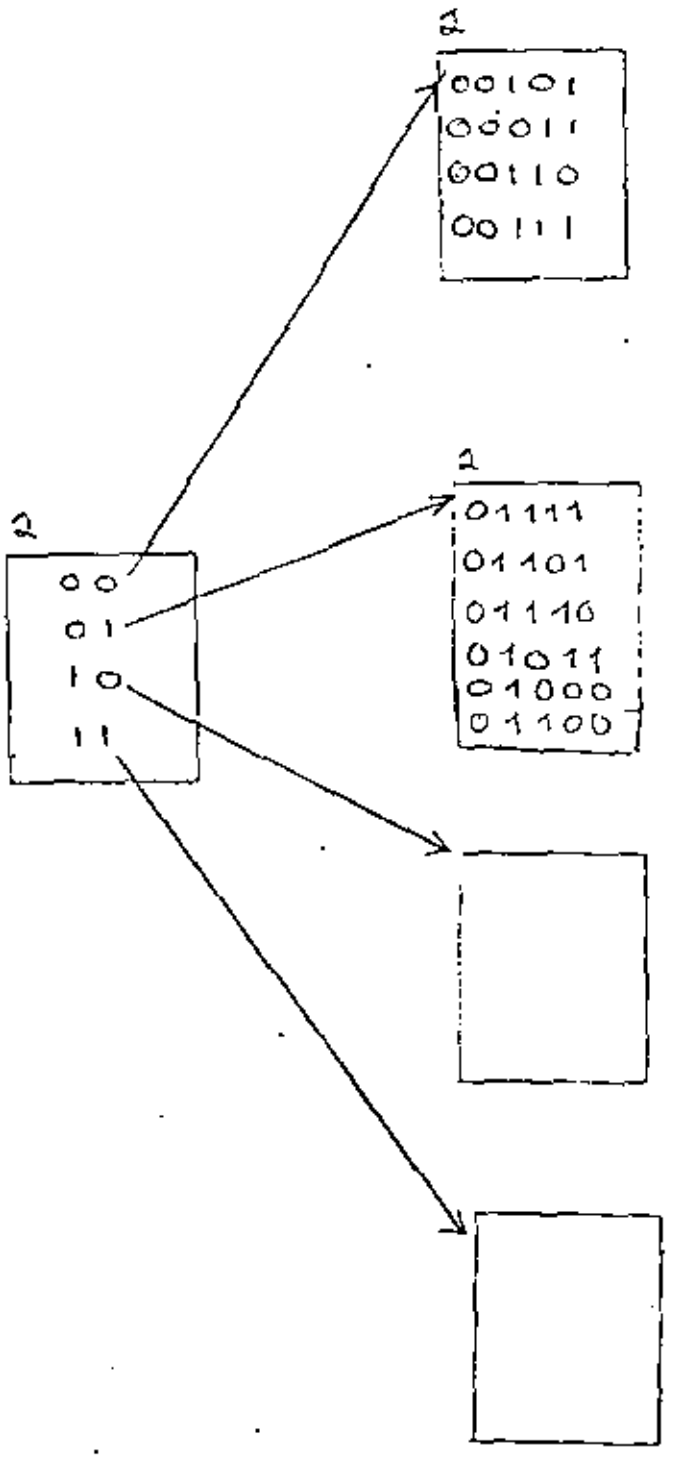
EN UN ESQUEMA DE DISPERSION TIPICO SE NECESITA UNA TABLA CON 2 147 483 647 ENTRADAS

NRO. DE REGISTRO	SEUDOLLAVE
1	00101 . . .
2	11000 . . .
3	11110 . . .
4	00110 . . .
5	10100 . . .
6	00011 . . .
7	11001 . . .
8	01111 . . .
9	00111 . . .
10	01101 . . .
11	01110 . . .
12	10001 . . .
13	01100 . . .
14	01011 . . .
15	10010 . . .
16	01000 . . .

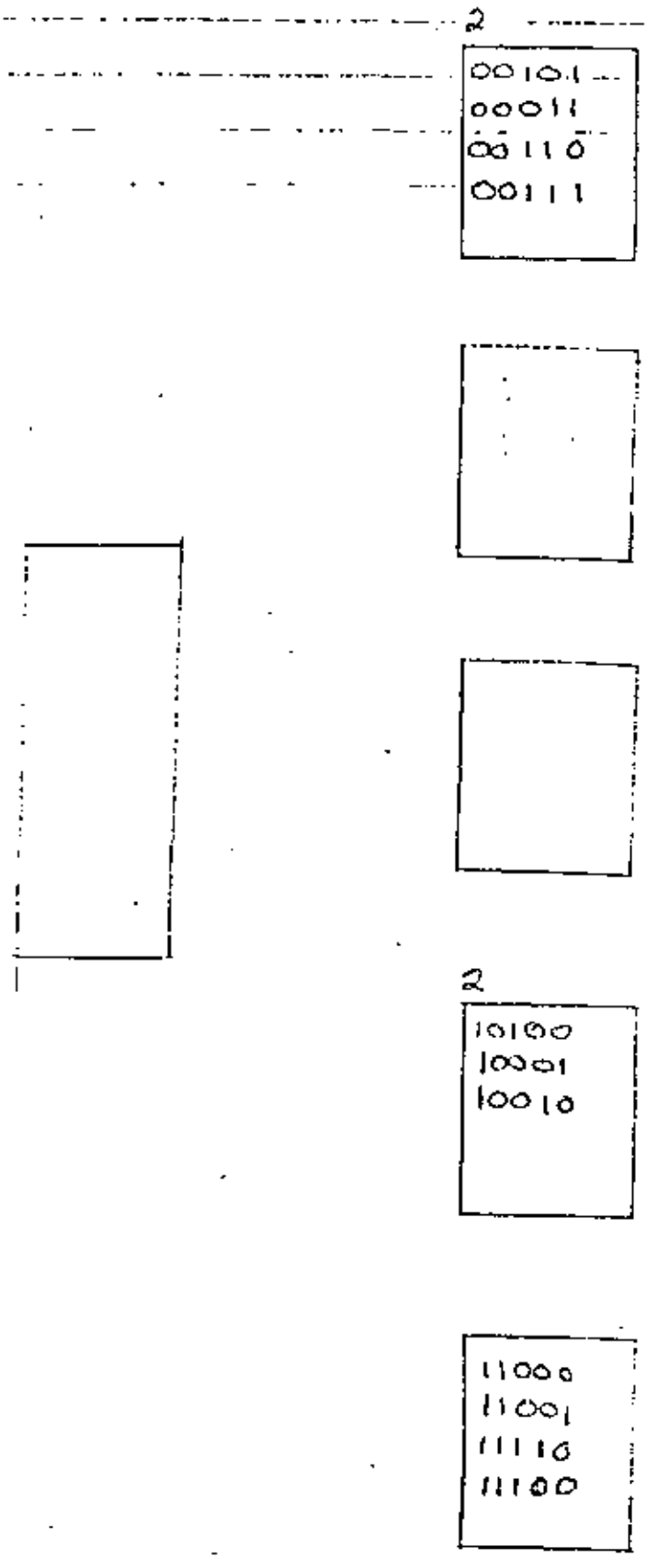


Supongamos que queremos insertar un registro con seudollave 00001...

Supongamos que queremos insertar un registro con una seudollave que comienza con "10" ó "11". Esto ocasiona que la tercer hoja se divida



Insertando un registro con seudollave 01001. Ocasiona que la segunda hoja se divida pero como su profundidad es 2 ahora su profundidad es 3


2
00101
00011
00110
00111
2
10100
10001
10010
11000
11001
11110
11100

Algoritmo BUSCAR. Busqueda en un archivo estructurado para dispersion extensible de un registro con una llave particular. EL algoritmo asume que las llaves son unicas.

LLAVE	Llave del registro a localizar
REG	Registro que contiene la llave
DIRECTO	Directorio
APTR	Campo apuntador de DIRECTO
SLLAVE	Valor de seudollave de la llave
D	Profundidad del directorio
I	Indice en el directorio

1.- Aplique la funcion de dispersion H a la llave

SLLAVE ← H(LLAVE)

2.- Tome los primeros D bits de SLLAVE y asignelos a en I

I ← primeros D bits de SLLAVE

3.- Tome el apuntador de DIRECTO, que señala a la pagina que contiene llaves que comienzan con I

APTR ← campo apuntador de DIRECTO[I]

4.- Busque en la pagina señalada por APTR el registro con llave LLAVE

5.- Si (el registro no esta) luego
Imprimir 'Registro con llave LLAVE no se encontro'

6.- Fin.

Algoritmo INSERTAR. Inserta un registro. Este algoritmo asume que todas las llaves para los registros son unicas

LLAVE
REG
DIRECTO
SLLAVE
D
I
ATR
NVOATR Apuntador a una nueva pagina si una division ocurre
PD Profundidad de la pagina

1.- Si (LLAVE y REG no ocasiona division de una pagina)
Copiar SLLAVE y REG en la pagina señalada por ATR
fin Salida.

2.- Obtener espacio para una nva pagina señalada por NVOATR

3.- Incrementar la profundidad de la pagina
PD ← PD + 1

4.- Coloque los registros en las paginas señaladas por ATR y NVOATR

5.- Si (la profundidad de la nueva pagina es mayor que la profundidad del directorio DIRECTO)
Incremente la profundidad del directorio en 1 : D ← D + 1
Duplique el tamaño del directorio y actualce los apuntadores
obien
Actualce el directorio de modo que apunte a las paginas apuntadas por ATR y NVOATR

6.- Fin fin

Algoritmo SUPRIMIR. Suprime un registro. Este algoritmo asume que todas las llaves de los registros son unicas

LLAVE
DIRECTO
SLLAVE
D
I
ATR
NVOATR
PD

1.- Ejecute algoritmo BUSCAR para obtener la direccion del registro que se suprime

2.- Si(no hay registro con llave LLAVE) luego
    Impresion 'no se encontro registro'
    Salida
fin

3.- Suprima el registro haciendo nulo el apuntador a el o haciendo su area de almacen blancos o nulo

4.- Si(la pagina que contiene el registro suprimido y las paginas arriba y abajo de esta pueden combinarse con factor de carga menor que el maximo deseado) luego
    Combinar las paginas y actualizar directorio
fin

5.- Si(cada apuntador en el directorio es igual que su compañero) luego
    Decrementar en 1 la profundidad del directorio y dividir el directorio
fin

6.- Fin

BIBLIOGRAFIA PARA DISPERSION PERFECTA

LEWIS, T. G. 1981 "SIMULATION OF PERFECT HASHING FUNCTION." REPORT, DEPARTMENT OF COMPUTER SCIENCE, OREGON STATE UNIVERSITY, CORVALLIS, OREGON

SPRUGNOLI, R. 1977." PERFECT HASHING FUNCTIONS: A SINGLE PROBE RETRIEVING METHOD FOR STATIC SETS." CACM 18, No. 11 (November), pp. 841-850.

The formatted output is then printed:

```
                                                  167 MAIN STREET
                                                  WINNIPEG 1, MANITOBA
                                                  AUGUST 17, 1975


MR. A.L. STRIDER
2014 CENTENNIAL DRIVE
THOMPSON, MANITOBA


DEAR MR. STRIDER,

     THE BUSINESS WORLD IS RAPIDLY CHANGING AND OUR CORPORATION HAS
BEEN KEEPING PACE WITH THE NEW REQUIREMENTS IN MID-CENTURY OFFICE
MACHINERY.  WE ARE GIVING YOU, MR. STRIDER, AS A KEY FIGURE IN THE
THOMPSON BUSINESS COMMUNITY, AN OPPORTUNITY TO BECOME FAMILIAR WITH
THE LATEST ADVANCEMENTS IN OUR EQUIPMENT.  A REPRESENTATIVE OF OUR
CORPORATION IN MANITOBA WILL BE SEEING YOU WITHIN THREE WEEKS.  HE
WILL TAKE SEVERAL MACHINES TO THOMPSON WHICH ARE INDICATIVE OF A WHOLE
NEW LINE OF OFFICE MACHINES WE HAVE RECENTLY DEVELOPED.
     OUR SALES REPRESENTATIVE IS LOOKING FORWARD TO HIS VISIT IN
THOMPSON.  HE FEELS THAT THE MACHINES HE SELLS COULD BECOME AN
INTEGRAL PART OF YOUR OFFICE ONLY A FEW DAYS AFTER INSTALLATION.


                                                  SINCERELY,


                                                  ROGER SMITH, MANAGER
                                                  OFFICE DEVICES CORPORATION
```

to recognize the reserved words of a source program. A language such as COBOL has several hundred reserved (key) words, so a hashing technique is almost essential in order for the compiler to determine quickly whether or not a particular word is a reserved word. Certain key words such as READ or ADD may initiate special processing, since they cause the generation of executable instructions. Others, such as VALUE, imply the initialization of data areas. In this section we will present several perfect hashing algorithms and hope that you may be motivated to try to describe other algorithms.

Sprugnoli (see References) gives two types of perfect hashing functions: the *quotient-reduction method* and the *remainder-reduction method*. The quotient-reduction method uses the formula

$$h(k) = \left\lfloor \frac{k + s}{N} \right\rfloor$$

where L means "truncate to the nearest integer," k is the key being hashed, and s and N are integers. For a given set of keys, the problem is to find s and N such that for every key, $h(k)$ is unique. In his paper, Sprugnoli gives an algorithm for determining s and N once the set of keys is known. Lewis (see References) shows ways of improving that algorithm and an alternative heuristic approach.

The remainder-reduction method hashing formula is

$$h(k) = \left\lfloor \frac{(d + kq) \bmod M}{N} \right\rfloor$$

where L means "truncate to the nearest integer," k is the key being hashed, and d, q, M, and N are integers. Sprugnoli also gives algorithms for finding the integer constants in the hashing formula. The remainder-reduction algorithm works well on keys that are not uniformly distributed. Taking $(d + kq)$ mod M helps scramble the keys and makes them more evenly distributed. The quotient-reduction method has no mod M operation and hence works better on uniformly distributed keys.

As an example showing a remainder-reduction hashing algorithm, we take the 12 months of the year, each given as its three-digit abbreviation. We think of the month in its character form: in storage, the characters can be used as binary numbers. Figure 8.11 lists the months and the decimal value of the second two characters of the abbreviation coded in EBCDIC. Including the first character in the value is not necessary, since there are only 12 keys and the extra character only makes the key value larger.

Sprugnoli gives the constants for the remainder-reduction algorithm of the 12 keys. The values are $d = 2304$, $q = 256$, $M = 23$, and $N = 2$. Taking the keys and performing the hashing algorithm yields the results in Figure 8.11. Note that this function yields values 0 through 11. The hash table is as small as possible: Its loading factor is 1.0.

Month	Decimal	Hash	Month	Decimal	Hash
JAN	49621	5	JUL	58579	10
FEB	50626	6	AUG	58567	4
MAR	49625	0	SEP	50647	3
APR	55257	7	OCT	50147	1
MAY	49640	11	NOV	55013	9
JUN	58581	2	DEC	50627	8

Figure 8.11 Hash values for twelve months

Another method for perfect hashing functions is given by Cichelli (see References). His hash function is independent of the character coding scheme (EBCDIC or ASCII) for a particular machine. Its formula is:

$$h(k) = \text{length of } k + \text{associated value of } k\text{'s first character} + \text{associated value of } k\text{'s last character}$$

where k is the key being hashed. For a particular set of keys, we must compute the associated values for the characters. We will not present that algorithm, but will leave it as a reference for the interested reader.

The approach when applied to Pascal's reserved word list can produce a hash table of size 36. Figure 8.12 lists the reserved words and their corresponding hash values. The characters' associated values used in the hash function are the following: A = 11, B = 15, C = 1, D = 0, E = 0, F = 15, G = 3, H = 15, I = 13, J = 0, K = 0, L = 15, M = 15, N = 13, O

Reserved Word	Hash Value	Reserved Word	Hash Value
do	2	record	20
end	3	packed	21
else	4	not	22
case	5	then	23
downto	6	procedure	24
goto	7	with	25
to	8	repeat	26
otherwise	9	var	27
type	10	in	28
while	11	array	29
const	12	if	30
div	13	nil	31
and	14	for	32
set	15	begin	33
or	16	until	34
of	17	label	35
mod	18	function	36
file	19	program	37

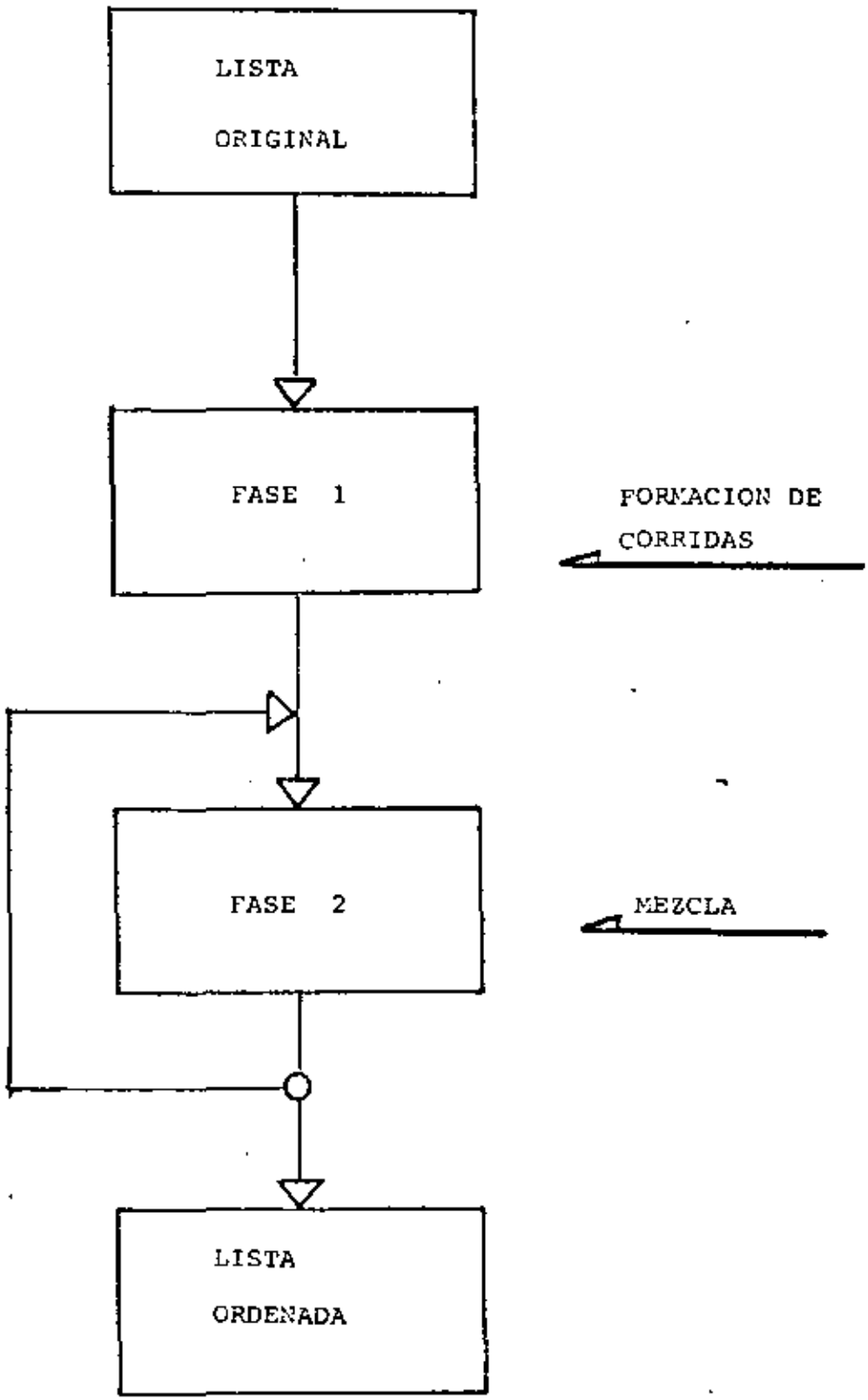
Figure 8.12 Pascal's reserved words and hash values

ESTRUCTURAS DE DATOS

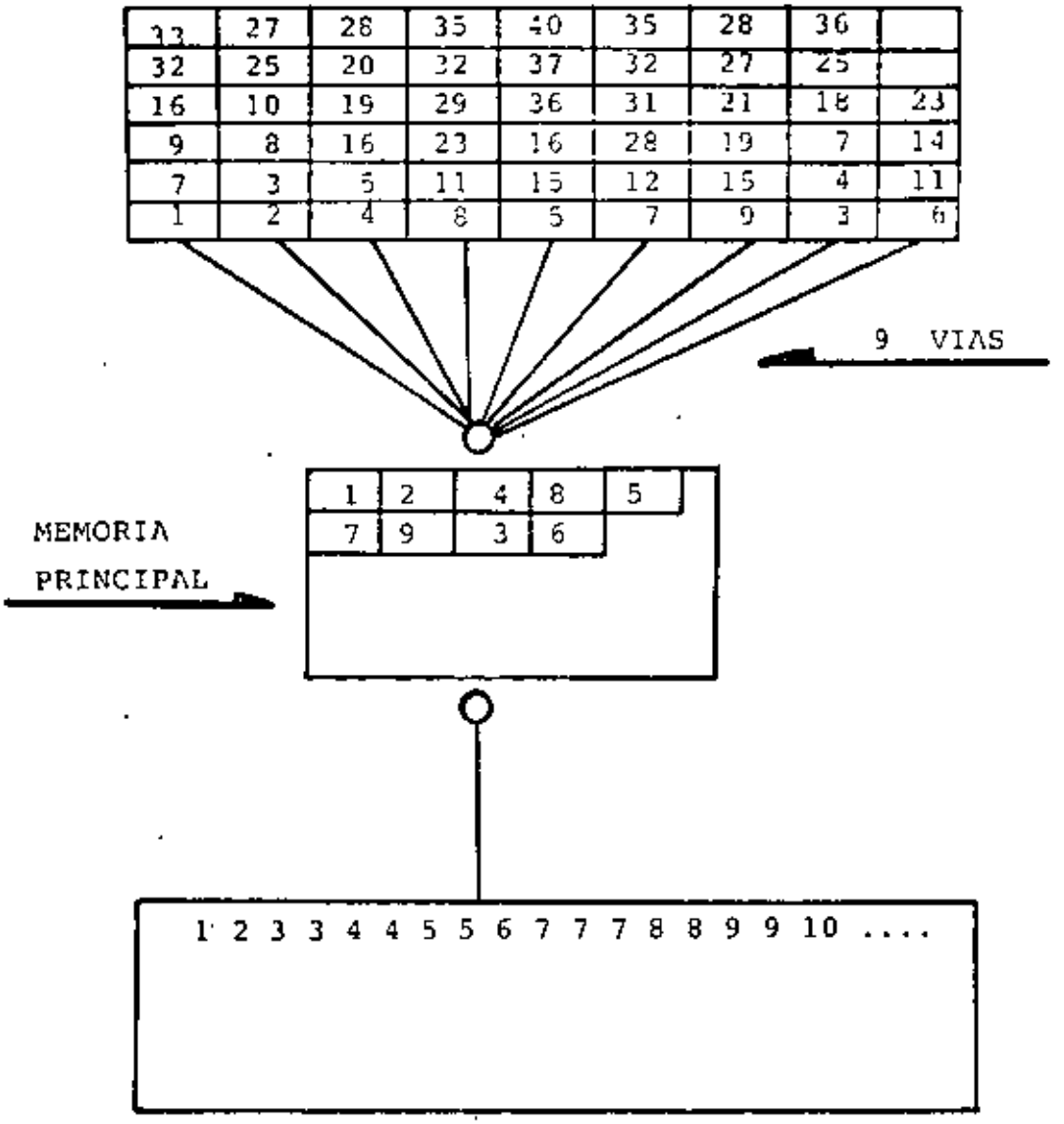
ORDENAMIENTOS EXTERNOS

M. EN. C. RICARDO CIRIA MERCE

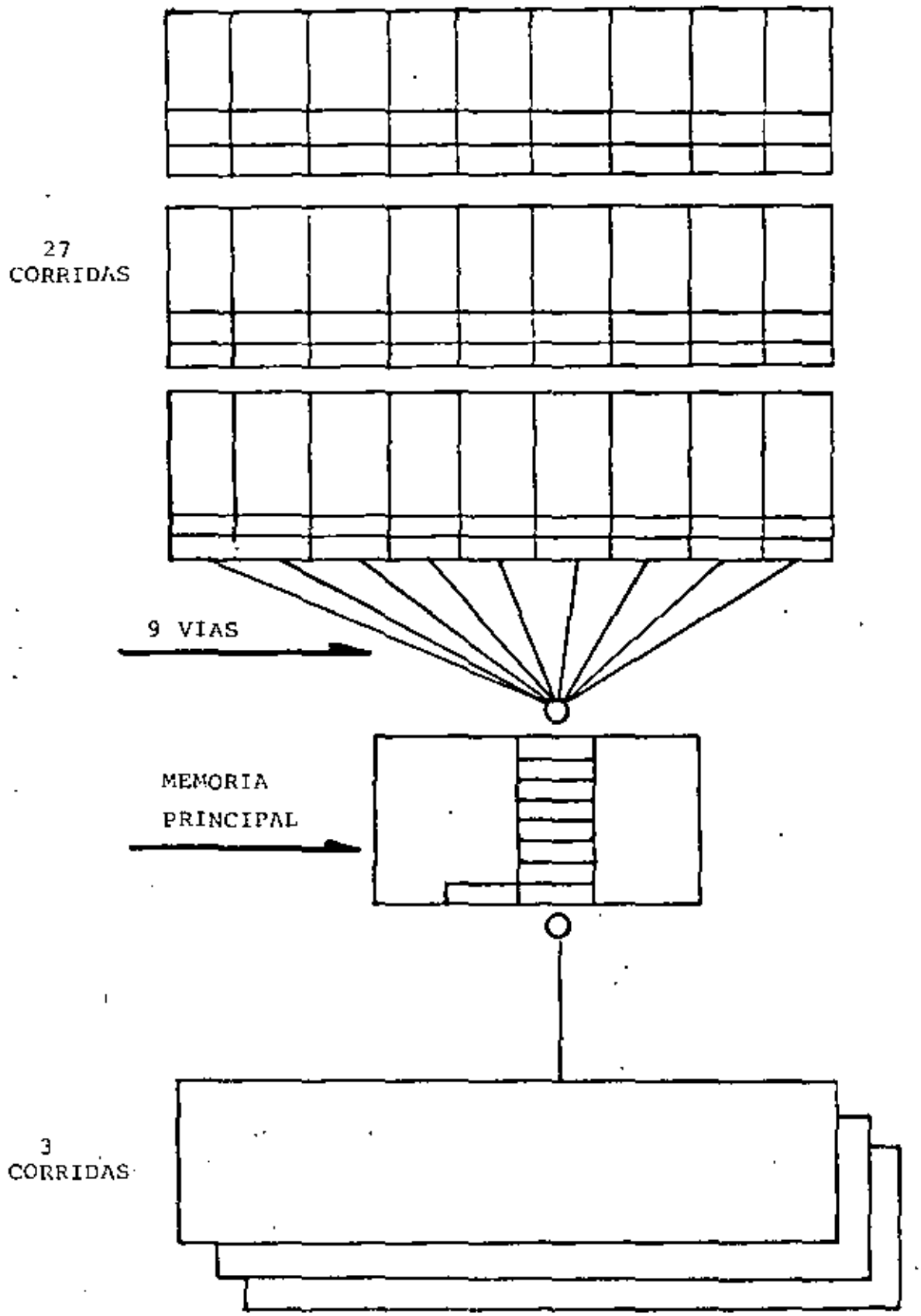
13 MAYO, 1983

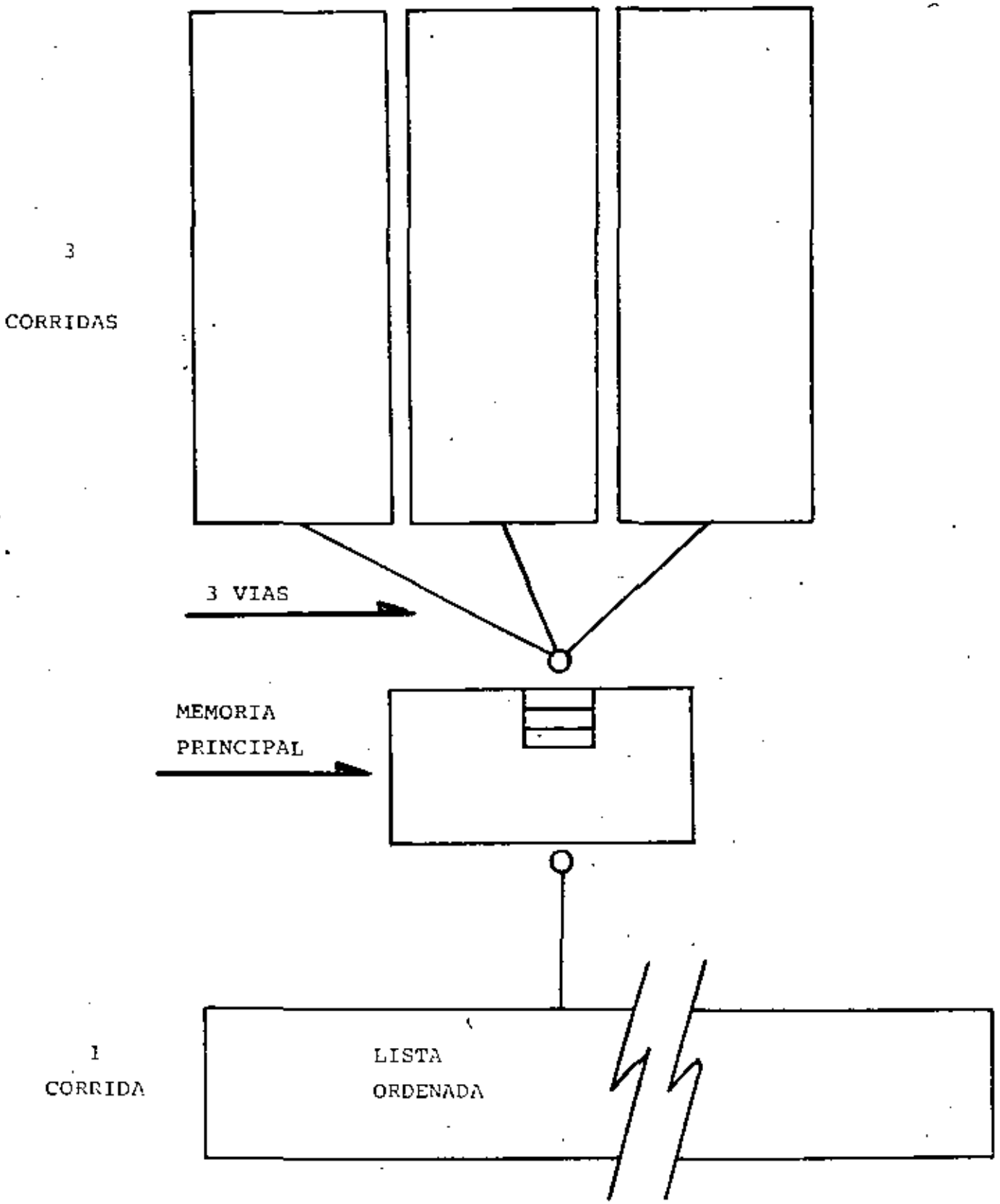

LISTA
ORIGINAL
FASE 1
FORMACION DE
CORRIDAS
FASE 2
MEZCLA
LISTA
ORDENADA

LISTA ORIGINAL FRACMANTADA EN 9 COPRIDAS :



LISTA ORDENADA


27
CORRIDAS
9 VIAS
MEMORIA
PRINCIPAL
3
CORRIDAS


3
CORRIDAS
3 VIAS
MEMORIA
PRINCIPAL
1
CORRIDA
LISTA
ORDENADA

FORMACION DE CORRIDAS

1) ORDENAMIENTO INTERNO

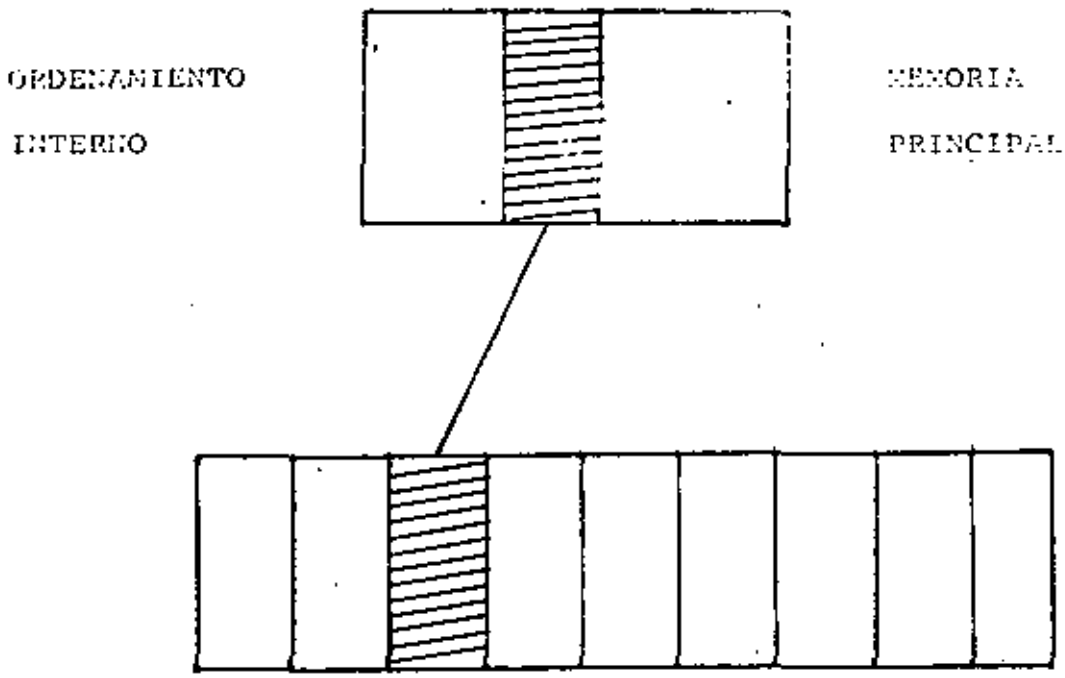
DE UNA FRACCION DE LA LISTA ORIGINAL

2) SELECCION DE CORRIDAS

EXISTENTES EN LA LISTA ORIGINAL

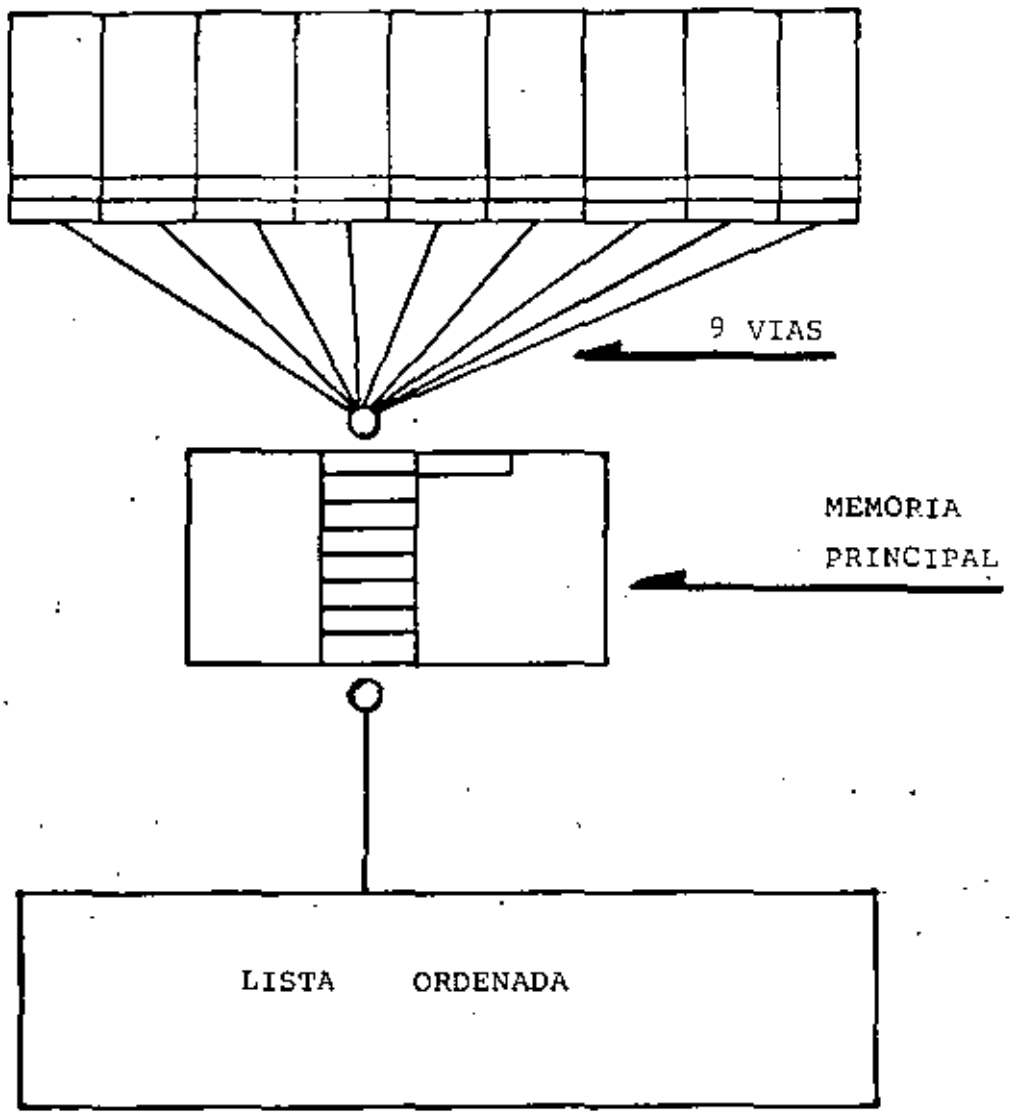
3) FORMACION DE CORRIDAS

MEDIANTE UN METODO PARA ESTE EFECTO



LISTA ORIGINAL FRAGMENTADA.

LISTA ORIGINAL FRAGMENTADA EN 9 CORRIDAS :



SELECCION DE CORRIDAS

ENTRADA :

6,9,2,5,7,3,1,4,8,.....

6,9/2,5,7/3/1,4,8,.....

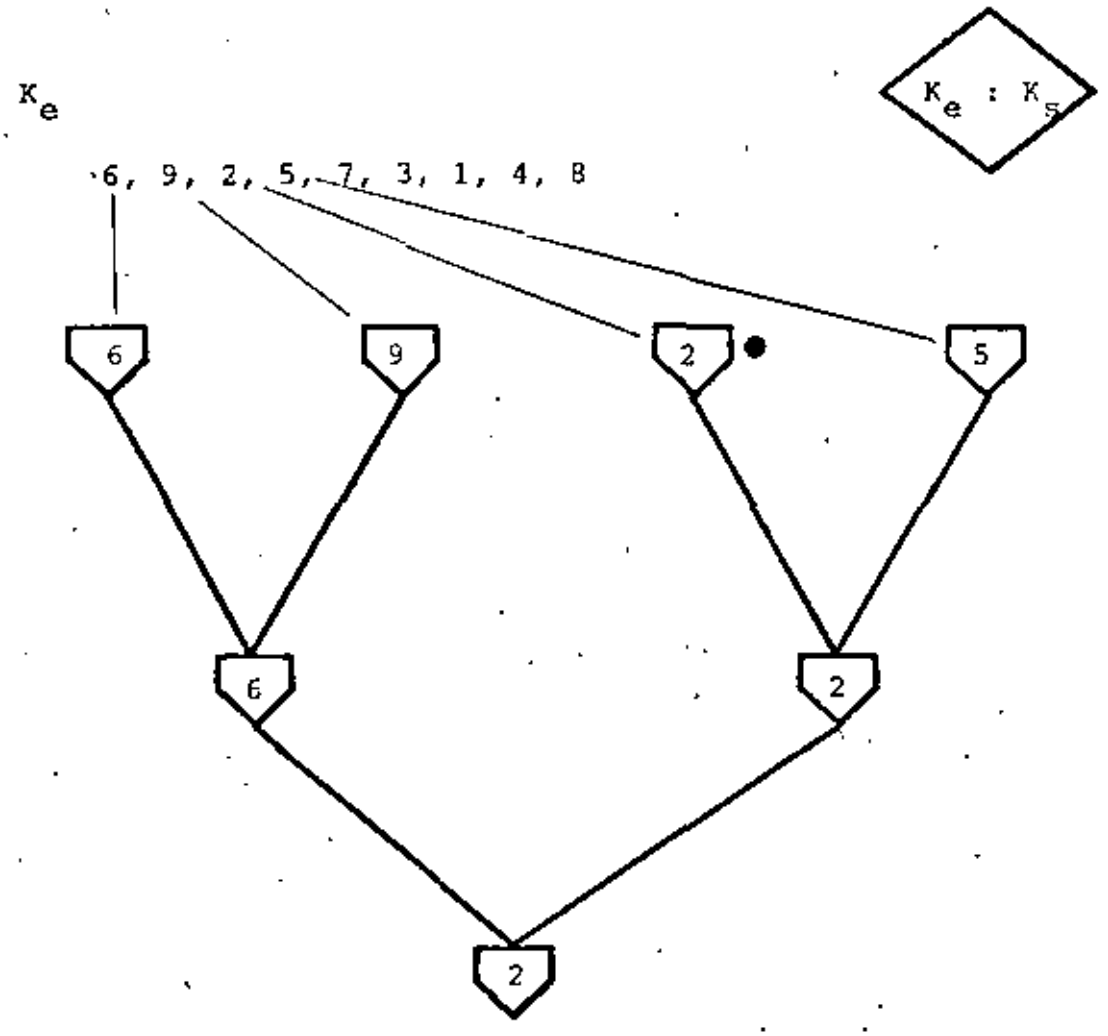
SALIDA :

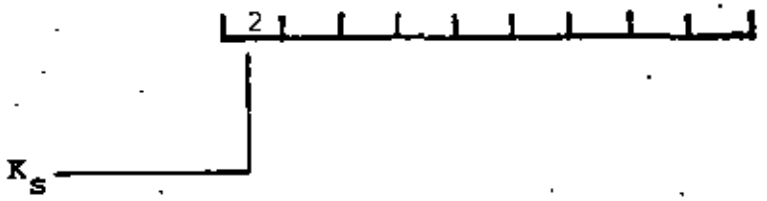
ACCESANDO LA LISTA SECUENCIALMENTE SE PONDRA UNA MARCA AL "ROMPERSE" EL ORDEN DE LA SUB_LISTA.

4 CORRIDAS

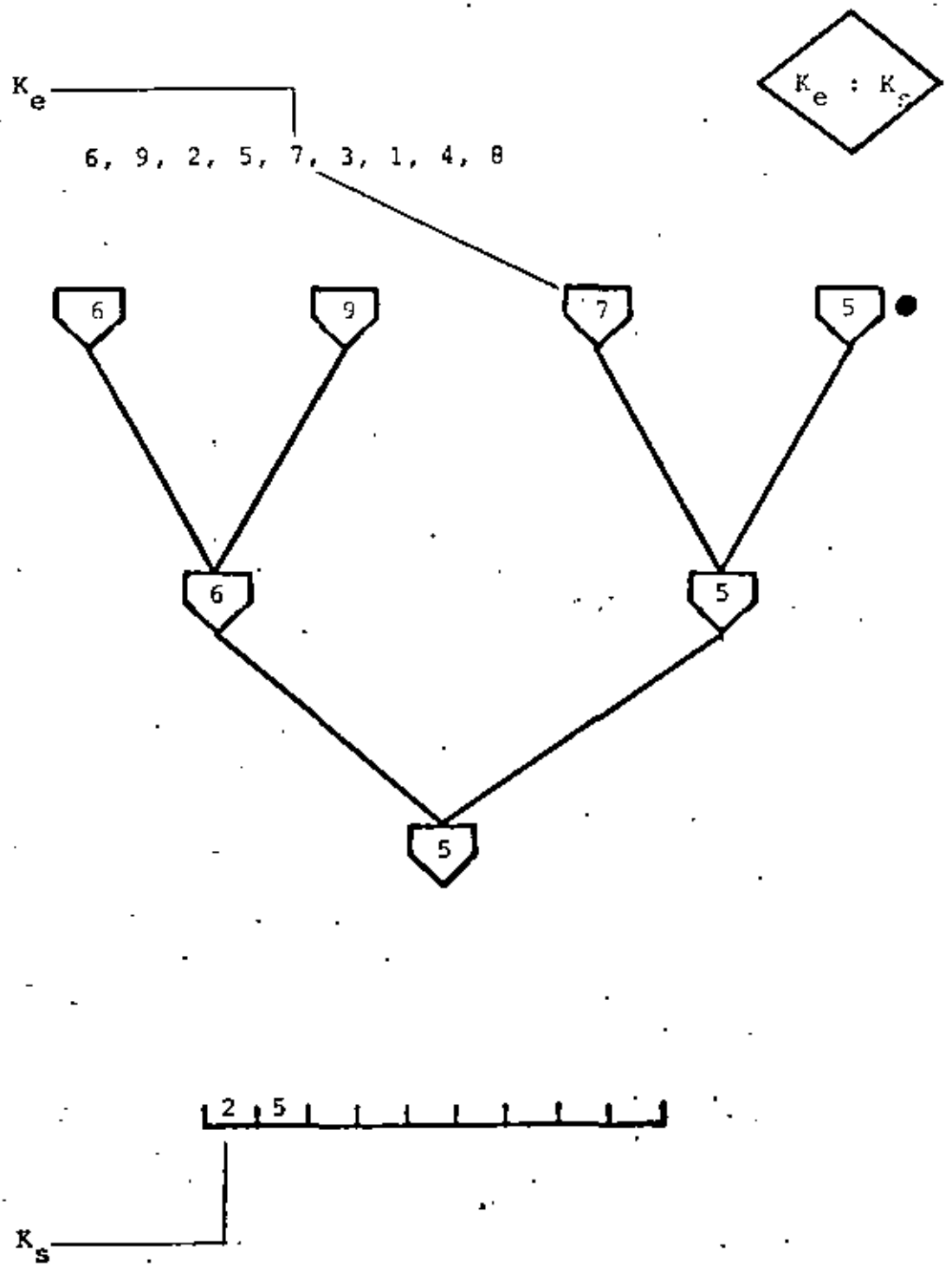
CORRIDAS CON LONGITUD MAXIMA DE 3 ELEMENTOS

FORMACION DE CORRIDAS

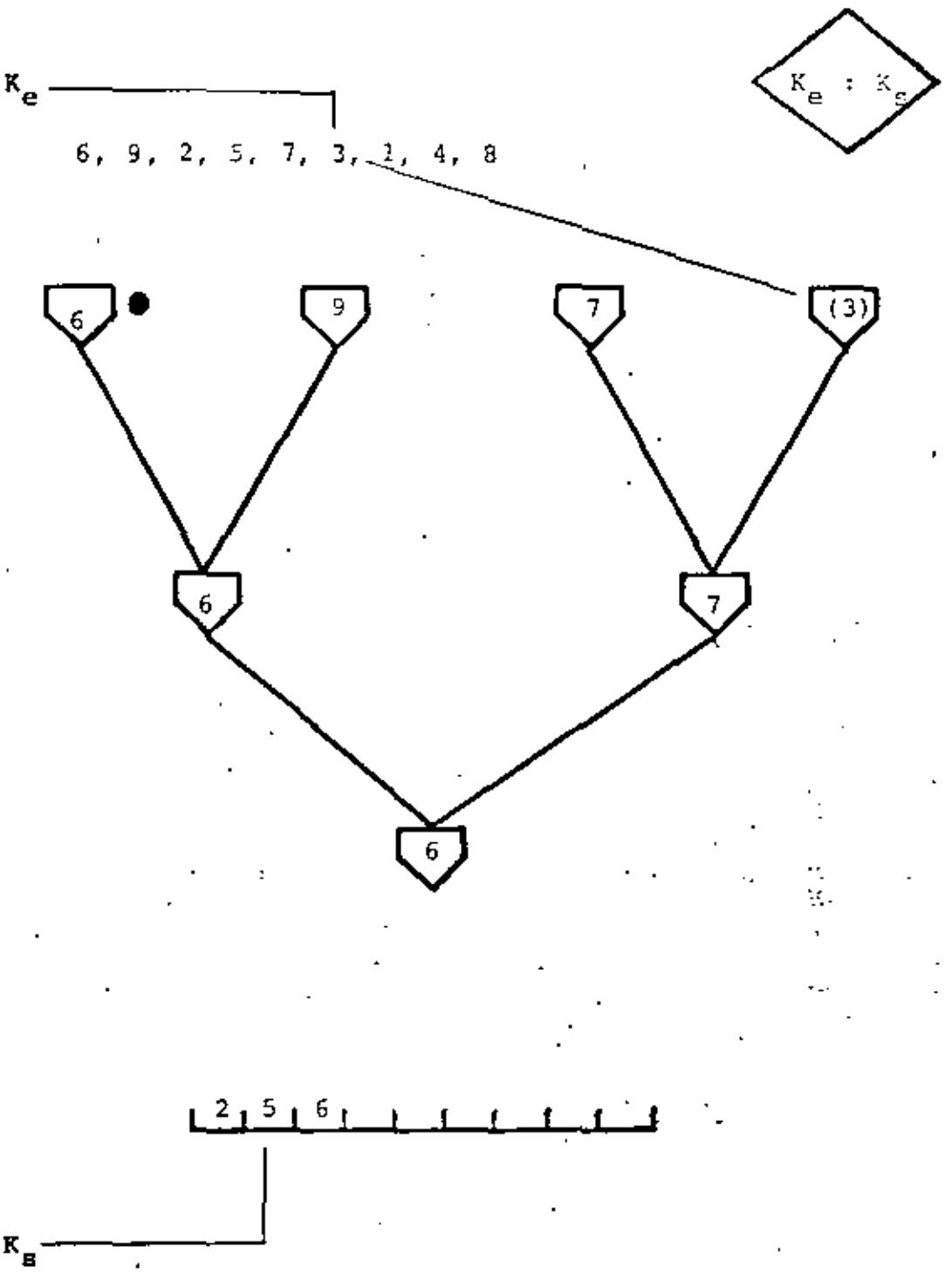




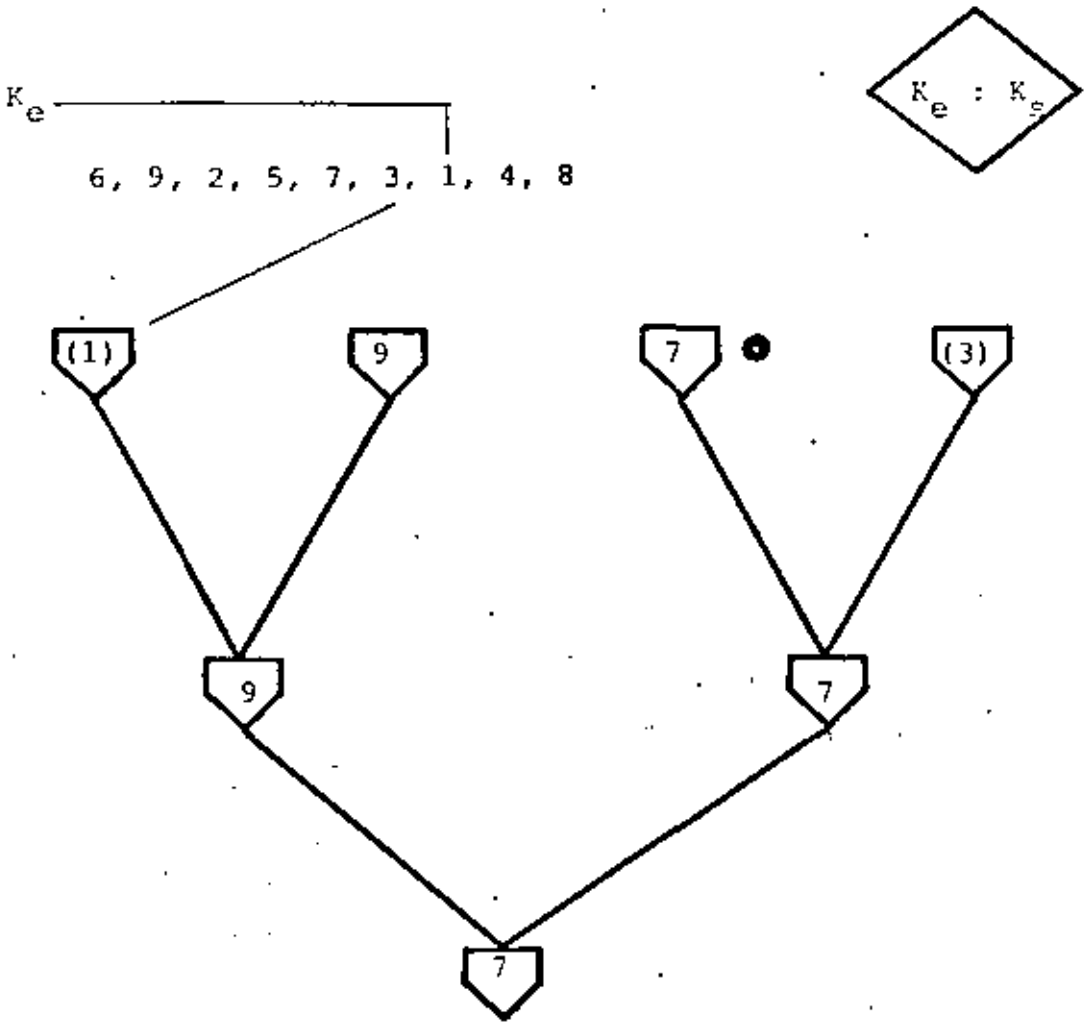
FORMACION DE CORRIDAS

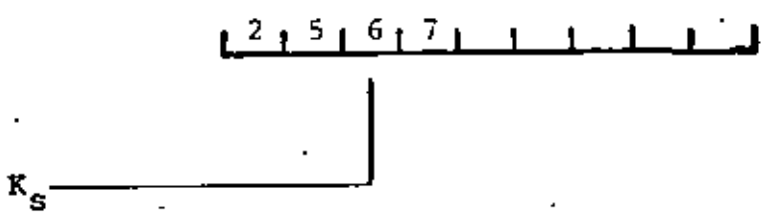


FORMACION DE CORRIDAS

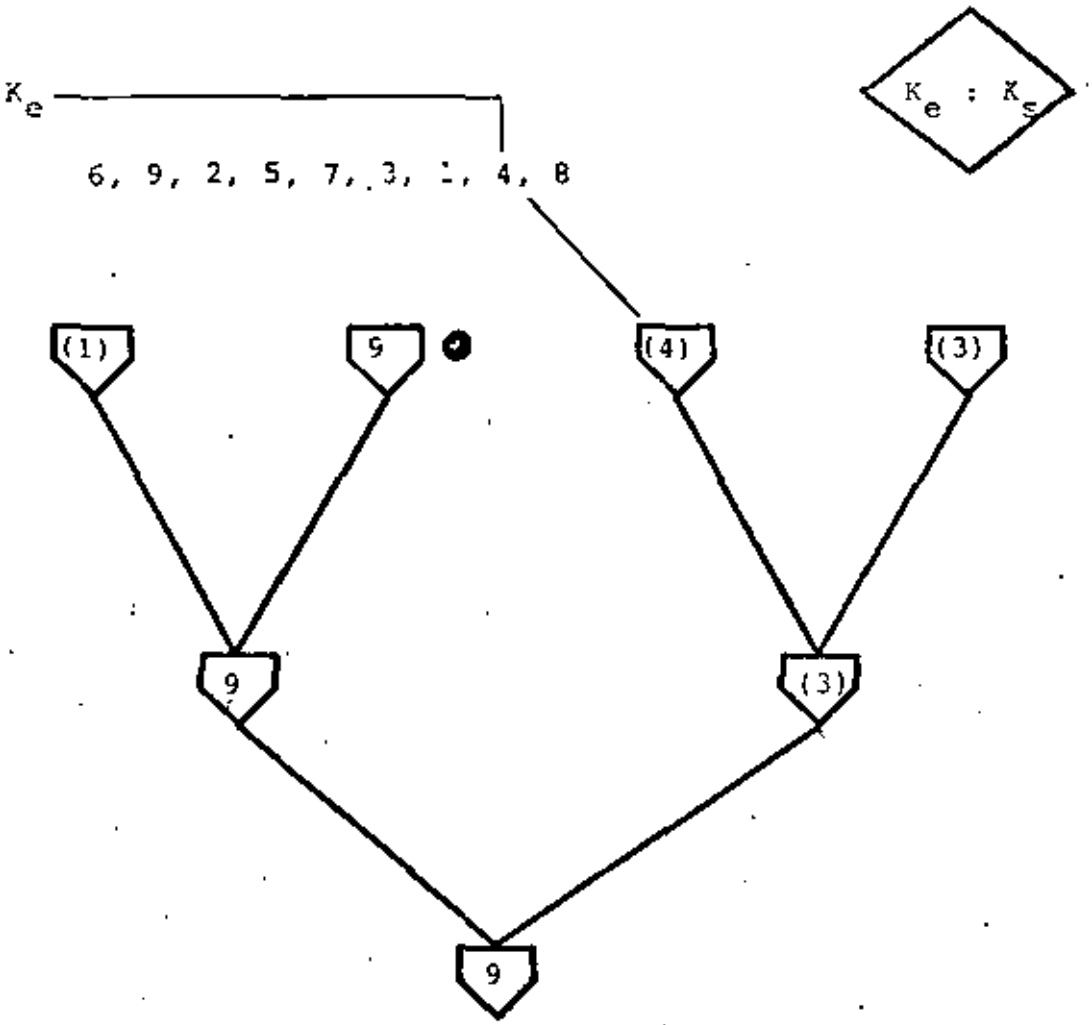


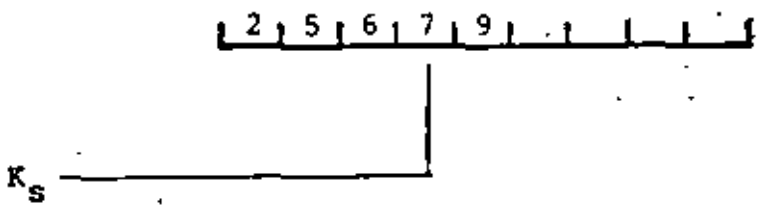
FORMACION DE CORRIDAS



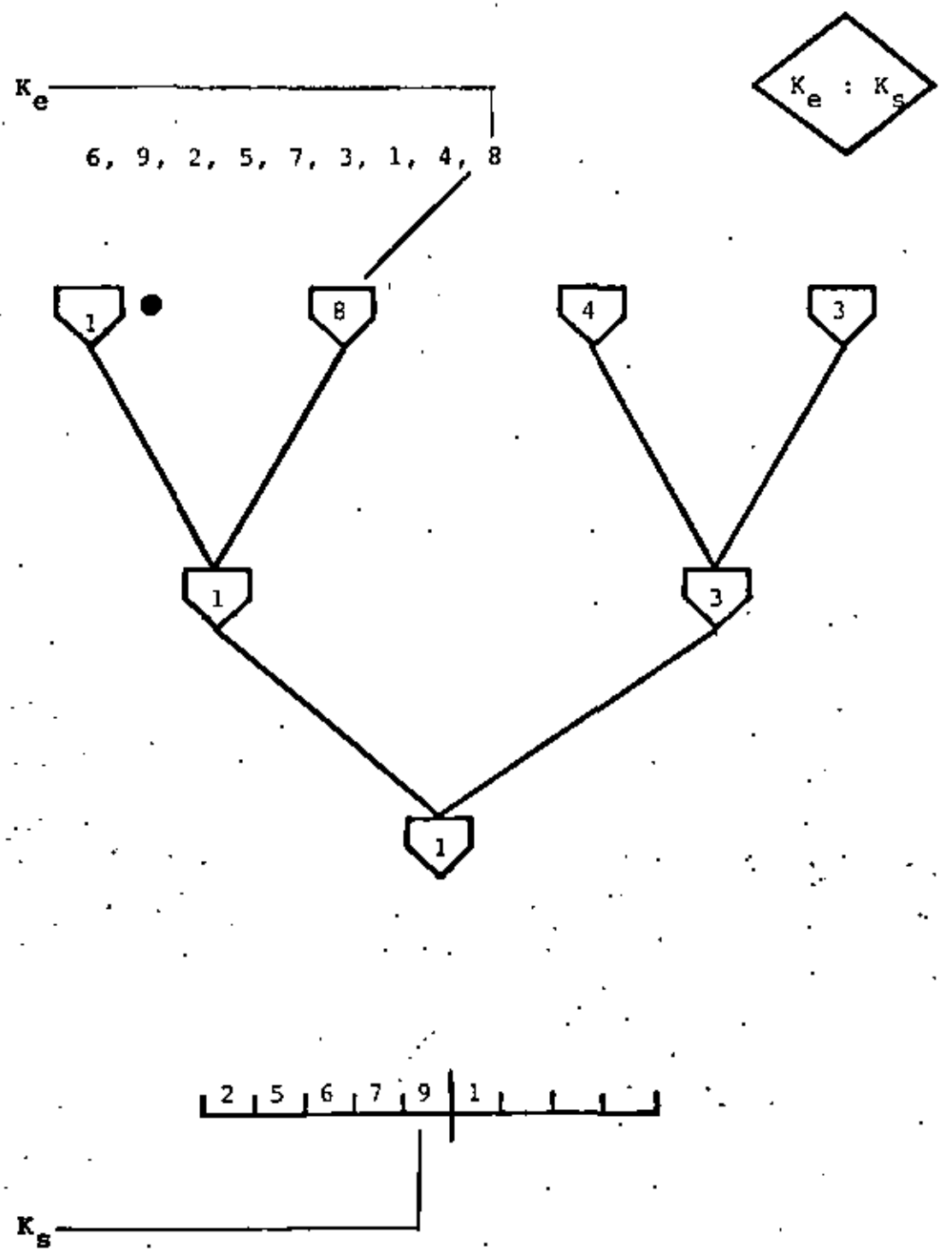


FORMACION DE CORRIDAS



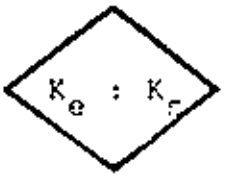


FORMACION DE CORRIDAS

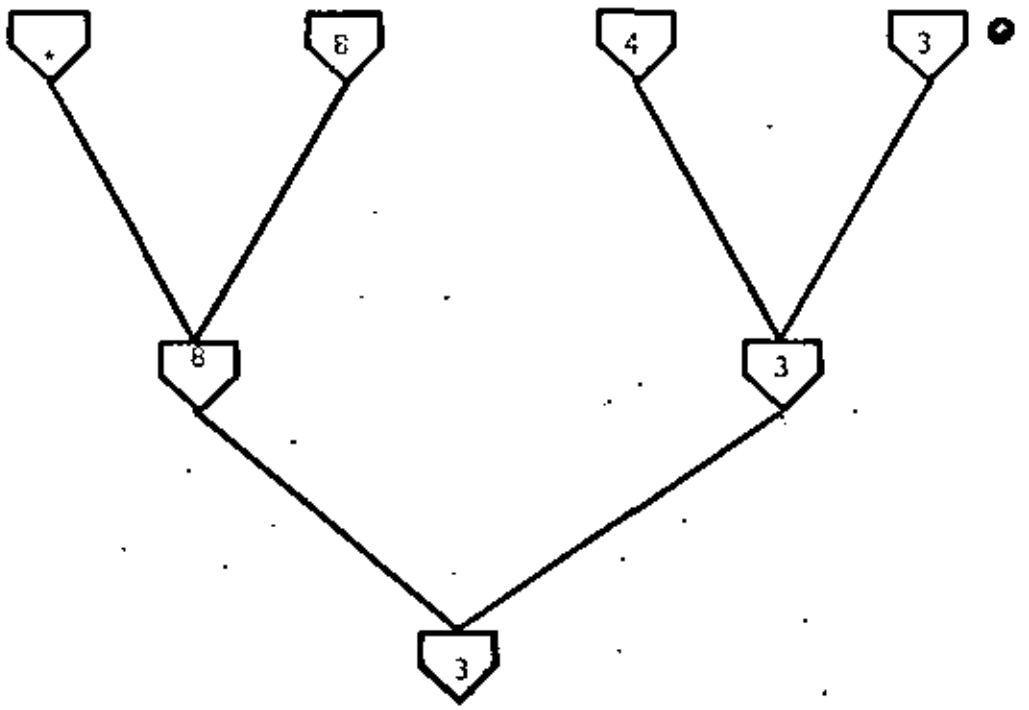


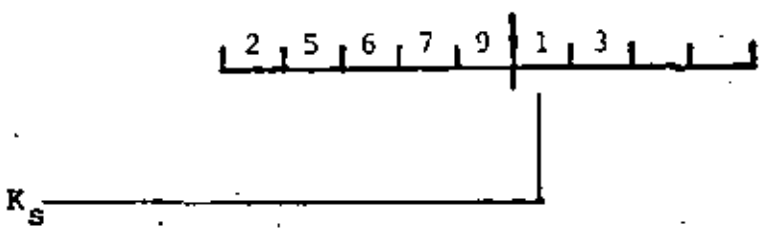
FORMACION DE CORRIDAS

K_e



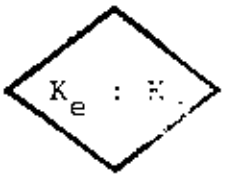
6, 9, 2, 5, 7, 3, 1, 4, 8



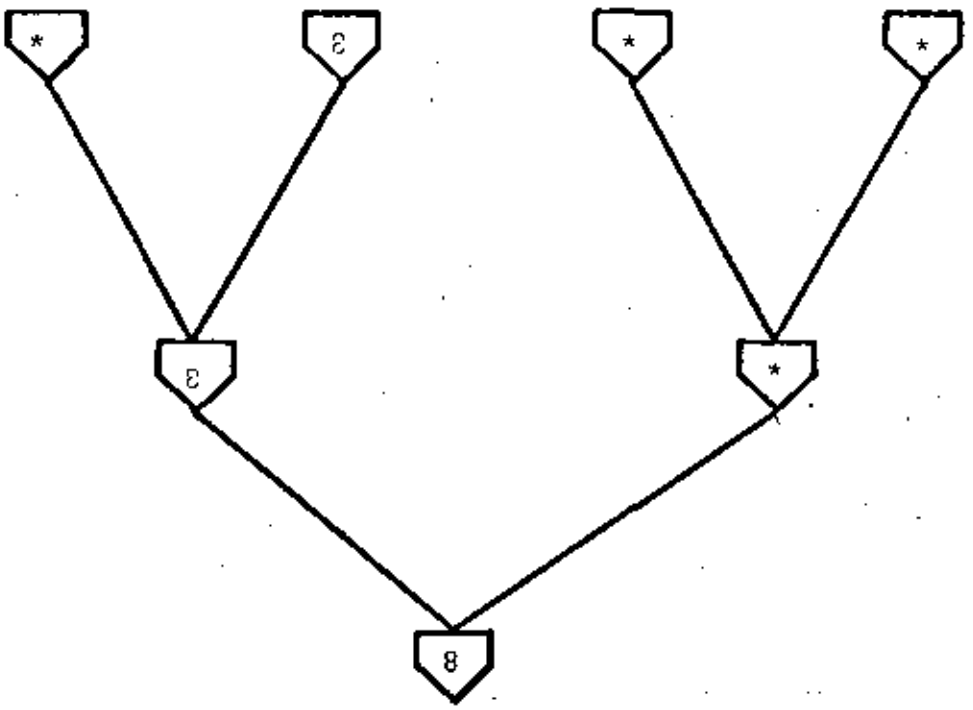


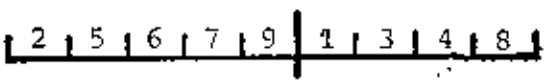
FORMACION DE CORRIDAS

K_e



6, 9, 2, 5, 7, 3, 1, 4, 8





K_s

2 CORRIDAS
LONGITUD MAXIMA DE 5 ELEMENTOS

POLIPHASE MERGE

SUPONGAMOS 3 UNIDADES DE CINTA Y 21 CORRIDAS DE LONGITUD RELATIVA 1.

N	C1	C2	C3
	1111111111(10)	11111111111(11)	---
1	---	1	222222222(10)
2	3	---	222222222
3	---	5	22222222
4	7	---	2222222
5	---	9	222222
6	11	---	22222
7	---	13	2222
8	15	---	222
9	---	17	22
10	19	---	2
11	---	21	---

POLIPHASE MERGE

N	C1	C2	C3
	11111111(8)	111111111111(13)	---
1	---.	11111(5)	2222222(8)
2	33333(5)	---	222(3)
3	33(2)	555(3)	---
4	---	5(1)	88(2)
5	13(1)	---	8(1)
6	---	21(1)	---

POLIPHASE MERGE

LEONARDO PISANO (LEONARDO DE PISA)

LEONARDO FIBONACCI (FILIUS BONACCII O HIJO DE BONACCIO)

AÑO DE 1202

"LIBER ABBACI" (LIBRO DEL ABACO)

0, 1, 1, 2, 3, 5, 8, 13, ...

$F_0 = 0 \qquad F_1 = 1 \qquad F_{n+2} = F_{n+1} + F_n \qquad n \geq 0$

$$F_n = \frac{1}{\sqrt{.5}} \cdot (\phi^n - \hat{\phi}^n)$$

$$\hat{\phi} = 1 - \phi = \frac{1}{2} (1 - \sqrt{5})$$

$\hat{\phi} = -0.61803$ y $\hat{\phi}^n$ es muy pequeño para n grandes

---

$$F_n = \frac{\phi^n}{\sqrt{5}}$$

$$\phi = 1 - \frac{1}{2}(1 - \sqrt{5}) = 1.61803\ 39887\ 49894\ 84802$$

POLIPHASE MERGE

NIVEL	C1	C2	TOTAL
0	1	0	1
1	1	1	2
2	2	1	3
3	3	2	5
4	5	3	8
5	8	5	13

PARA 6 CINTAS :

NIVEL	C1	C2	C3	C4	C5	TOTAL
0	1	0	0	0	0	1
1	1	1	1	1	1	5
2	2	2	2	2	1	9
3	4	4	4	3	2	17
5	8	8	7	6	4	44
6	16	15	14	12	8	65
n+1	C1+C2	C1+C3	C1+C4	C1+C5	C1	

EXTERNAL RADIX SORT

SUPONIENDO QUE SOLO HAY LLAVES :

000	001	010	011	100	101	110	111
0	1	2	3	4	5	6	7

y la siguiente lista :

3, 7, 0, 2, 5, 1, 6, 4

CON 4 UNIDADES DE CINTA.

C1	C2	C3	C4
37025164	---	---	---
---	---	0264	3751
0451	2637	---	---
---	---	0123	4567

CON 6 UNIDADES DE CINTA, PODEMOS UTILIZAR RADIX 3, PUDIENDO ORDENAR LLAVES DESDE

$$0 \text{ hasta } 3^k - 1$$

EN k PASADAS.

ESTE ORDENAMIENTO ES, GENERALMENTE, INFERIOR A LOS QUE UTILIZAN MEZCLAS.

EXTERNAL RADIX SORT

SUPONIENDO QUE SOLO HAY LLAVES :

000	001	010	011	100	101	110	111
0	1	2	3	4	5	6	7

y la siguiente lista :

3, 7, 0, 2, 5, 1, 6, 4

CON 4 UNIDADES DE CINTA.

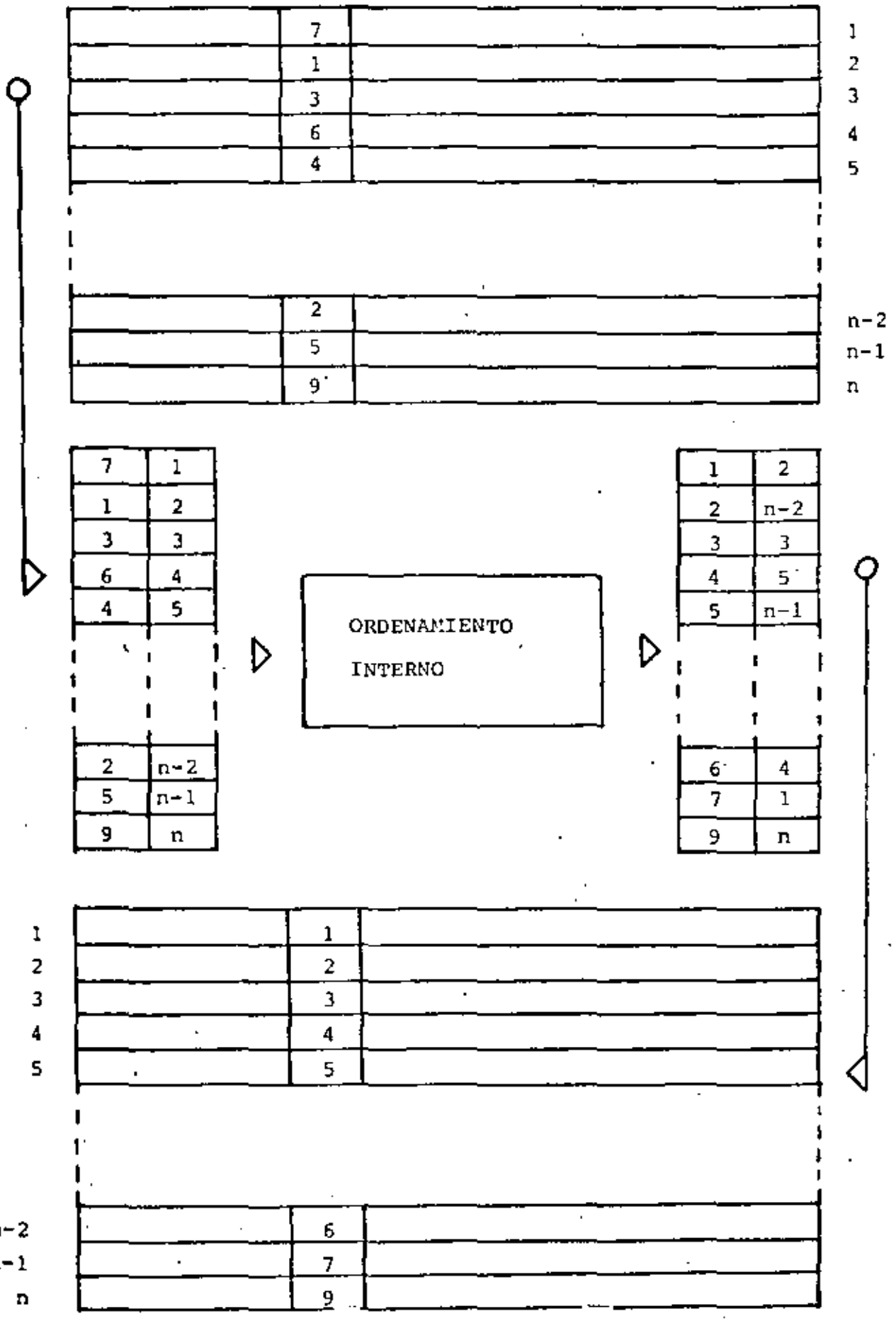
C1	C2	C3	C4
37025164	---	---	---
---	---	0264	3751
0451	2637	---	---
---	---	0123	4567

CON 6 UNIDADES DE CINTA, PODEMOS UTILIZAR RADIX 3, PUDIENDO ORDENAR LLAVES DESDE

0 hasta $3^k - 1$

EN k PASADAS.

ESTE ORDENAMIENTO ES, GENERALMENTE, INFERIOR A LOS QUE UTILIZAN MEZCLAS.


7
1
3
6
4
2
5
9
1
2
3
4
5
n-2
n-1
n
ORDENAMIENTO
INTERNO

MERGE CASCADA

NOTACION: n (M)

M: Longitud relativa de una corrida

n: Número de corridas.

Supongamos que contamos con 6 unidades y la distribución original de 190 corridas es la siguiente:

C 1	C 2	C 3	C 4	C 5	C 6
55(1)	50(1)	41(1)	29(1)	15(1)	—
40(1)	35(1)	26(1)	14(1)	—	15(5)
26(1)	21(1)	12(1)	—	14(4)	
14(1)	9(1)	—	12(3)		
5(1)	—	9(2)			
—	5(1)	9(2)	12(3)	14(4)	15(5)

MERGE CASCADA

C 1	C 2	C 3	C 4	C 5	C 6
—	5(1)	9(2)	12(3)	14(4)	15(5)
5(15)	—	4(2)	7(3)	9(4)	10(5)
	4(14)	—	3(3)	5(4)	6(5)
		3(12)	—	2(4)	3(5)
			2(9)	—	

5(15)	4(14)	3(12)	2(9)	1(5)	—
4(15)	3(14)	2(12)	1(9)	—	1(55)
3(15)	2(14)	1(12)	—	1(50)	
2(15)	1(14)	—	1(41)		
1(15)	—	1(29)			

—	1(15)	1(29)	1(41)	1(50)	1(55)
1(190)	—	—	—	—	—

MERGE CASCADA

C 1	C 2	C 3	C 4	C 5	TOTAL
55	50	41	29	15	190
15	14	12	9	5	55
5	4	3	2	1	15
1	1	1	1	1	5

		ACTUAL		ANTERIOR
C_5	=		+	C_1
C_4	=	C_5	+	C_2
C_3	=	C_4	+	C_3
C_2	=	C_3	+	C_4
C_1	=	C_2	+	C_5

EN GENERAL:

Cn actual = Cl Anterior

Ck actual = Ck+1 Actual + Cn-K+1 Anterior

MERGE CASCADA

(5 CINTAS)

	C 1	C 2	C 3	C 4	TOTAL
1	1	1	1	1	4
2	1	2	3	4	10
3	4	7	9	10	30
4	10	19	26	30	85
5	30	56	75	85	246
6	85	160	210	246	701
7	246	456	616	701	2019
8	701	1317	1773	2019	5810
9	2019	3792	5109	5810	16730

SORT OSCILANTE

(Requiere unidades con capacidad de leer al revés) Sheldom Sobel (1962)

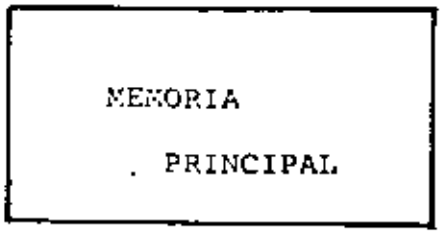
NOTACION

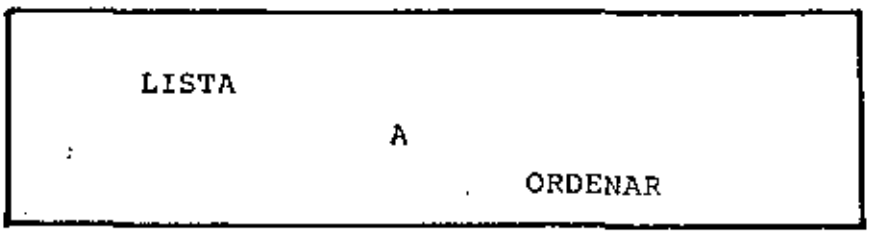
A (n) : Corrida ascendente de longitud n.

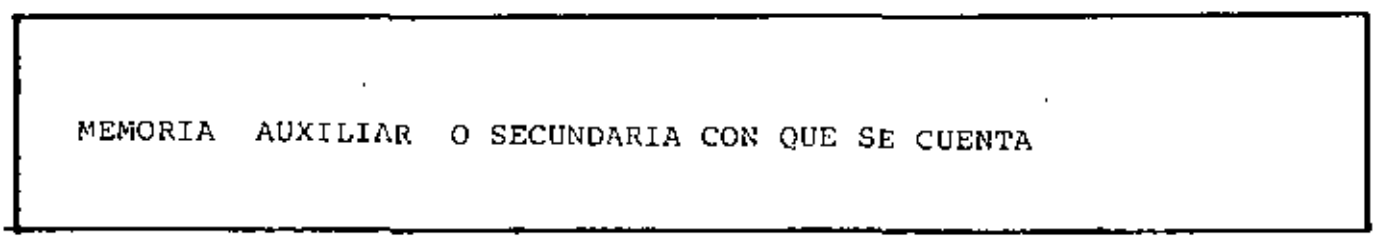
D (n) : Corrida descendente de longitud n.

Supongamos que contamos con 5 unidades de cinta y 16 corridas de longitud relativa 1.

	OPERACION	C_1	C_2	C_3	C_4	C_5
1	DISTRIBUCION	A(1)	A(1)	A(1)	A(1)	—
2	MEZCLA	—	—	—	—	D(4)
3	DISTRIBUCION	—	A(1)	A(1)	A(1)	D(4) A(1)
4	MEZCLA	D(4)	—	—	—	D(4)
5	DISTRIBUCION	D(4)A(1)	—	A(1)	A(1)	D(4) A(1)
6	MEZCLA	D(4)	D(4)	—	—	D(4)
7	DISTRIBUCION	D(4)A(1)	D(4)A(1)	—	A(1)	D(4) A(1)
8	MEZCLA	D(4)	D(4)	D(4)	—	D(4)
9	MEZCLA	—	—	—	A(16)	—


MEMORIA
PRINCIPAL


LISTA
A
ORDENAR


MEMORIA AUXILIAR O SECUNDARIA CON QUE SE CUENTA

DIRECTORIO DE ASISTENTES AL CURSO DE ESTRUCTURAS DE DATOS (DEL 15 DE ABRIL AL 13 DE MAYO DE 1983)

NOMBRE Y DIRECCION	EMPRESA Y DIRECCION
1. SANDRA CAPETILLO TERRERO Monte Albán 518-101-A Deleg. Benito Juárez C.P. 03600 México, D. F. 6 19 51 34	SECRETARIA DE COMUNICACIONES Y TRANSPORTES Xola y Av. Universidad Col. Narvarte Deleg. Benito Juárez México, D. F. 6 72 30 46
2. HECTOR CASTRO BAUTISTA Av. Cuauhtémoc 1026 Col. Narvarte Deleg. Benito Juárez México, D. F.	INSTITUTO MEXICANO DEL PETROLEO Eje Central Lázaro Cárdeas No. 152 Deleg. Gustavo A. Madero C.P. 07730 México, D. F. 5 67 66 00
3. LUIS CASTRO HERRERA	A. C. NIELSEN J. L. Lagrange No. 103 Ch. Morales Deleg. M. Hidalgo México, D. F. 3 95 03 99
4. JAVIER RAMIREZ BARRAGAN Moras No. 762 Col. del Valle Deleg. Benito Juárez C.P. 03100 México, D. F. 5 34 91 10	TRITURADOS BASALTICOS DEPRIVADOS Bosque de Ciruelos No. 130 Bosques de las Lomas Deleg. Miguel Hidalgo C.P. 11700 México, D. F. 5 96 56 33
5. RAFAEL DE LA CRUZ GONZALEZ Av. 16 de Septiembre No. 45 Xochimilco, D. F. 5 50 52 15 Ext. 3639	INSTITUTO DE INGENIERIA Y FACULTAD DE INGENIERIA, UNAM Av. Insurgentes Universidad y Copilco
6. EDMUNDO ETCHEBURY ALVAREZ Providencia 325-303 Col. del Valle Z.P. 12 México, D. F. 6 87 24 87	INSTITUTO POLITECNICO NACIONAL Unidad Profesional Zacatenco Edificio 8-3er. Piso México, D. F. 5 86 32 07

DIRECTORIO DE ASISTENTES AL CURSO DE ESTRUCTURAS DE DATOS (DEL 15 DE ABRIL AL 13 DE MAYO DE 1983)

NOMBRE Y DIRECCION	EMPRESA Y DIRECCION
7. JOSE LUIS DE LA VEGA ESTRADA Lerdo de Tejada "D" No. 615 Unidad Tlatelolco Deleg. Cuauhtémoc México, D. F. 5 83 87 69	DESARROLLO DE INGENIERIA, S. A. de C.V. (DISA) Pablo de la Llave No. 110 Bosques de Tetlameya Deleg. Coyoacán C.P. 04730 México, D. F. 5 73 12 03
8. ELVIRA GARCIA FLORES	CERAMICA SANTA JULIA, S. A. Insurgentes Sur No. 1605 San José Insurgentes México, D. F. 5 34 80 20
9. GABRIEL GOMEZ FAMILA Convento Yuriria No. 23 Tlanepantla, Edo. de México 3 94 10 83	DESARROLLO DE INGENIERIA, S. A. Pablo de la Llave 110 C.P. 4730 México, D. F. 5 73 12 03
10. JOSE GONZALEZ GALICIA Calle 43 No. 66 Col. I. Zaragoza Deleg. V. Carranza México, D. F. 7 62 05 39	A. C. NIELSEN J. L. Lagrange 103 Col. Ch. Morales Deleg. M. Hidalgo México, D. F. 3 95 03 99
11. FABIAN H. HERNANDEZ ARELLANO	A. C. NIELSEN J. L. Lagrange 103 Col. Ch. Morales Deleg. M. Hidalgo México, D. F. 3 95 03 99
12. ALEJANDRO JIMENEZ Andador Ahuejotes No. 6 Col. San Marcos Deleg. Xochimilco C.P. 16050 México, D F. 5 83 81 34	INSTITUTO DE INGENIERIA, UNAM Ciudad Universitaria México, D. F. 5 50 52 15 Ext. 3640

DIRECTORIO DE ASISTENTES AL CURSO DE ESTRUCTURAS DE DATOS (DEL 15 DE ABRIL AL 13 DE MAYO DE 1983)

NOMBRE Y DIRECCION	EMPRESA Y DIRECCION
13. GREGORIO LINARES URENDA Mineros Metalurgicos No. 259 Col. Monte Alto C.P. 02650 México, D. F.	DATABASE, S. C. Insurgentes Sur No. 634-305 Col. del Valle Deleg. Benito Juárez México, D. F. 5 23 89 93
14. RAUL LOPEZ CHAVEZ Sur 75 A No. 204 No. 4 Sinatel Deleg. Ixtapalapa C.P. 13 México, D. F. 6 70 16 17	INSTITUTO MEXICANO DEL PETROLEO Eje Central Lazaro Cárdenas No. 152 San Bartolo Atenehuacan Z.P. 14 México, D F. 5 67 66 00 Ext. 2657
15. JAIME ANTONIO MACHUCA GONZALEZ Bosques de Chihuahua 56 Santa Mónica Atizapán, Edo. de México C.P. 54050 México, D. F. 5 67 70 22	PRODUCTOS ESPECIALIZADOS DE ACERO, S. A. Poniente 134 No. 854 Col. Ind. Vallejo Azcapotzalco C.P. 02300 México, D. F. 3 97 11 97
16. RICARDO NAVA HERNANDEZ Bosques de Pakistan No. 91 Bosques de Aragón Edo. de México	INDUSTRIAS RESISTOL, S. A. Bosques de Ciruelos No. 99 Bosques de las Lomas Deleg. Miguel Hidalgo México, D. F. 5 96 35 88
17. JUAN ANTONIO NAVARRO MARTINEZ Impacto No. 286 Col. Atlanta Cuautitlan Izcalli, Edo. de México	FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN, UNAM Cuautitlan Edo. de México 3 31 15
18. ABEL NUÑEZ GUTIERREZ E. Pugibet No. 12 Centro México, D. F.	TELEFONOS DE MEXICO, S. A. E. Pugibet No. 12 Centro México, D. F. 5 85 34 44 Ext. 9703
19. JUAN CARLOS ORTEGA GUERRERO Edificio E-19 Depto. 32 Torres de Mixcoac Deleg. Alvaro Obregón México, D. F. 6 80 39 27	COMPER Indianápoles No. 4-201 Col. Napoles Deleg. Benito Juárez México, D. F. 5 43 11 94

DIRECTORIO DE ASISTENTES AL CURSO DE ESTRUCTURAS DE DATOS (DEL 15 DE ABRIL AL 13 DE MAYO DE 1983)

NOMBRE Y DIRECCION	EMPRESA Y DIRECCION
20. JESUS ANTONIO PATIÑO RAMIREZ Lago Muritz No. 85 Col. Anáhuac Deleg. Miguel Hidalgo C.P. 11320	FACULTAD DE INGENIERIA, UNAM Ciudad Universitaria San Angel México, D. F. 5 50 52 15 Ext. 4611
21. SALVADOR PEREZ VI RAMONTES Sur 77 No. 216 Col. Sinatel Iztapalapa C.P. 09470 México, D. F. 5 81 08 04	FACULTAD DE INGENIERIA, UNAM Ciudad Universitaria México, D. F. 5 50 52 15
22. ADRIANA RAMIREZ ESPINOSA Av. Central Mza. 331 Lot. 26 Cd. Azteca Edo. de México C.P. 15120	INDUSTRIAS RESISTOL, S. A. Bosques de Ciruelos No. 99 Bosques de las Lomas Deleg. Miguel Hidalgo México, D. F. 5 96 35 88
23. GUSTAVO RODRIGUEZ ORTIZ Tenis 157 H. 201 Country Club Coyoacán C.P. 04220 México, D. F. 5 44 50 81	D.I.S.A. Pablo de la Llave 110 Bosques de Tetlameya Coyoacán C.P. 04730 México, D. F. 5 73 12 03
24. GILBERTO SANTOS ARAOZ Av. Bordo No. 16 Sta. Ursula Coapa Deleg. Coyoacán México, D. F. 5 19 51 34	SECRETARIA DE COMUNICACIONES Y TRANSPORTES Xola y Av. Universidad Col. Narvarte Deleg. Benito Juárez C.P. 03600 México, D. F. 5 19 51 34
25. MARTHA SENTIES ARZAMENDI MARTHA Via Lactea 141 Prado Churubusco Coyoacán México, D. F. 5 82 26 43	DIRECCION GENERAL DE ESTADISTICA S.P.P. Insurgentes Sur 795 México, D. F. 6 87 29 11 ext. 168

DIRECTORIO DE ASISTENTES AL CURSO DE ESTRUCTURAS DE DATOS (DEL 15 DE ABRIL AL 13 DE MAYO DE 1983)

NOMBRE Y DIRECCION	EMPRESA Y DIRECCION
26. WALDO SOBERON FLORES Prol. Moctezuma 130 Col. Romero de Terreros Coyoacán 5 54 25 04	UNIVERSIDAD AUTONOMA CHAPINGO Chapingo, México 5 85 45 55 Ext. 5083
27. JOSE I. VALLE GARCIA Andador 28 del Temoluco No. 32 Casa 2 Acueducto de Guadalupe Gustavo A. Madero C.P. 07270 México, D. F.	INSTITUTO MEXICANO DEL PETROLEO Av. Eje Central Lazaro Cárdenas No. 152 San Bartolo Atepehuacan Deleg. Gustavo A. Madero México, D.F. 5 67 66 00 Ext. 2504