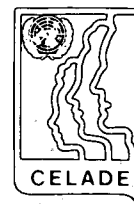


3224 (7922) c1

Centro Latinoamericano de Demografía

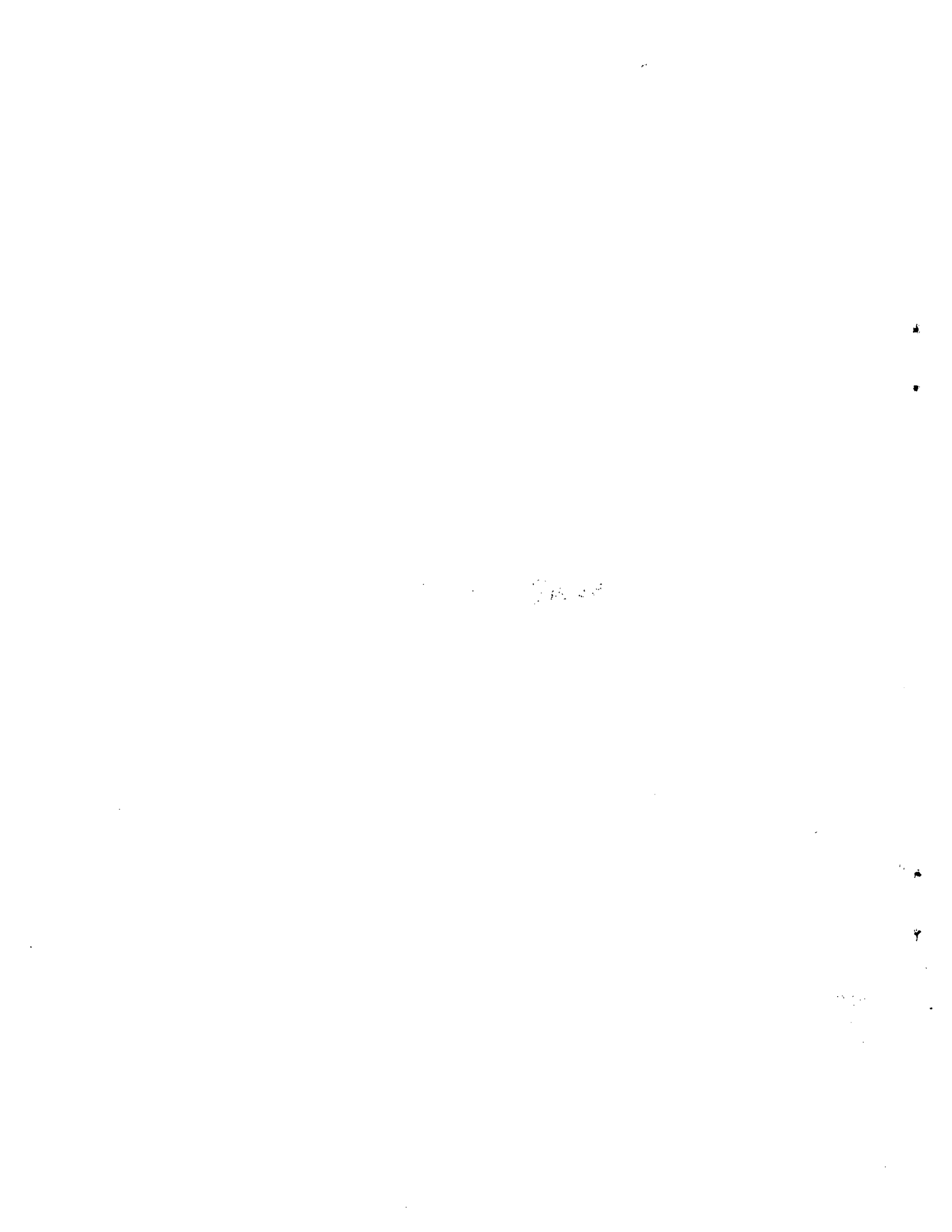


Documentos de Seminarios



PROGRAMA CONCOR

DS/8
150.
1975.



INDICE

	<u>Página</u>
Capítulo I: INTRODUCCION Y VISION GENERAL DEL SISTEMA	
1. Conceptos Involucrados	
a) Consistencia de rango	2
b) Consistencia entre variables.....	2
c) Corrección manual de los errores detectados	3
d) Corrección automática de los errores detectados	3
e) Obtención de un archivo de salida	3
2. Programas que Conforman el Sistema	
a) SUPERVISOR	3
b) ANALIZADOR	4
c) EXECUTOR	4
d) LISTER	5
e) CORRECTOR	5
Capítulo II: DESCRIPCION DEL LENGUAJE	
A. Restricciones del archivo de entrada	8
B. Tipos de sentencias	8
C. Codificación de las sentencias	9
D. Uso de constantes	10
E. Uso de variables	10
F. Uso del asterisco como valor de asignación.	10
G. Instrucciones del lenguaje	11
1. Instrucciones para definir archivo de entrada e inicializar variables	11
DICT	11
FILT	14
INIT	15
2. Instrucciones para el manejo del flujo de ejecución	16
GOTO	16
IF	17
STOP	18

	<u>Página</u>
3. Instrucciones de recodificación	18
XRC	18
SRC	19
DRC	21
4. Instrucciones de cálculo	22
CPT	22
5. Instrucciones de verificación	24
RANGE	24
IFL	26
DIFL	28
VERIF	30
6. Instrucciones para asignación diná- mica	32
DMN	32
STORE	32
ASSGN	33
ACT	34
7. Transformación de variables alfanumé- ricas a numéricas	36
TRANS	36
8. Definición de variables	37
SET	37
9. Instrucciones de salida	38
WRT	38
PUT	39
PUTF	40
10. Diccionario de errores	40
DICTE	41
END	42
11. Instrucciones al analizador	44
SPACE	44
EJECT	45
NOP	45
12. Instrucción al monitor	46
\$OPTION	46

	<u>Página</u>
Capítulo III: PROGRAMA CORRECTOR	
A. Explicación General	49
B. Archivos de Input/output y sus Restricciones	49
1. Archivo de entrada	49
2. Archivo de salida	50
C. Instrucciones para el Corrector	50
REPLACE	50
DELETE	51
INSERT	51
CHANGE	52

THE UNIVERSITY OF CHICAGO

PHYSICS DEPARTMENT

PHYSICS 311

LECTURE 10

CAPITULO I

Introducción y Visión General del Sistema

Hace aproximadamente un año, CELADE comenzó a diseñar y desarrollar un lenguaje orientado a la limpieza de datos provenientes de censos de población y encuestas. Para ser más precisos, el lenguaje en primera instancia tenía un decidido enfoque hacia la consistencia y corrección de datos censales y sólo hace unos tres meses que se incorporaron las instrucciones que permiten manejar con facilidad complejas relaciones entre variables, lo que habilita al lenguaje para el procesamiento de encuestas.

Al iniciarse el desarrollo del sistema, se tenían ideas muy claras sobre las facilidades que debía ofrecer al usuario para llevar a cabo la consistencia y corrección de sus datos, facilidades que involucran conceptos básicos conocidos por todos los productores de estadísticas y descritos en prácticamente todos los artículos sobre el tema. Sin embargo, a pesar de ser conceptos claros y por todos conocidos, el software desarrollado en este campo es sumamente escaso y se puede afirmar que hasta hace muy poco tiempo no había un solo lenguaje orientado o paquete que permitiera atender a todos los aspectos involucrados en la consistencia y corrección de datos, ya sea de encuestas o censos.

1. Conceptos Involucrados

A continuación se describirán brevemente los principales conceptos involucrados en la consistencia, ya que son ellos los que definen casi en su totalidad las características de este lenguaje orientado.

a) Consistencia de rango: entendiéndose por tal la verificación de que no haya información omitida o códigos o valores no definidos. En particular, en el caso de variables cuantitativas no es tan fácil limitar el intervalo "válido" y se toma un valor extremo considerado como máximo o mínimo razonable.

b) Consistencia entre variables: con frecuencia se da el caso de que una variable, aun teniendo códigos válidos o valores razonables, al confrontarla con otra u otras variables, manifiestan claramente una inconsistencia entre ellas. Por ejemplo, una mujer de 16 años de edad, que re-

registra 10 hijos nacidos vivos, revela un error evidente en una de las variables: edad o fecundidad. Sin embargo, ambas variables, al ser consideradas en forma individual, registran valores perfectamente admisibles.

c) Corrección manual de los errores detectados: cuando los datos en proceso provienen de una encuesta en que el número de casos es relativamente pequeño, el usuario puede desear corregir los errores detectados en forma manual. Esto es, que el computador le permita identificar el o los cuestionarios y la o las variables con conflictos para que él, mediante el estudio o el análisis del documento, realice las correcciones necesarias.

d) Corrección automática de los errores detectados: en otras ocasiones, ya sea porque el número de casos es muy elevado (censos u otras fuentes) o porque el método de imputación es demasiado complejo, el usuario puede desear que el computador corrija el error en forma automática siguiendo una metodología que él especificará.

e) Obtención de un archivo de salida exento de errores del tipo de los especificados en los puntos a) y b) en formato binario o punto flotante.

2. Programas que Conforman el Sistema

Para poder llevar a cabo la consistencia de datos, CONCOR dispone de 5 programas (véase la figura 1) que se describen a continuación:

a) SUPERVISOR: El usuario tendrá acceso a cualquiera de los cuatro programas restantes a través del supervisor, el cual espera encontrar como primera tarjeta del programa (C1) la especificación de la o las fases de consistencia que se desea ejecutar. (OPTION). De acuerdo con el contenido de dicha tarjeta, el supervisor cargará en la memoria principal los programas que se especifiquen de a uno por vez. El término normal de un programa da paso a la ejecución del próximo solicitado por el usuario en la tarjeta OPTION.

b) ANALIZADOR: Como se puede apreciar fácilmente en la figura 1, es prácticamente el corazón del lenguaje. El recibe todas las especificaciones de la consistencia de los datos del usuario por medio de sentencias o instrucciones (C2) regidas por reglas de sintaxis que se detallarán junto con la descripción de las posibles operaciones (véase el capítulo 2). A su vez, el analizador genera archivos que servirán de input para EXECUTOR y LISTER (D1 y D2 respectivamente).

Exceptuando las tarjetas de corrección de los cuestionarios (C3), el usuario solamente tiene comunicación directa con el analizador. De aquí que, pese a que la ejecución de los programas que componen el sistema es optativo, el analizador debe haber sido ejecutado por lo menos una vez a fin de crear D1 y D2.

El analizador produce un listado de las tarjetas que constituyen el programa del usuario, acompañado de un diagnóstico de errores de sintaxis. Adicionalmente, y en forma opcional, puede entregar una referencia completa de variables y rótulos en la cual se indican el o los números de sentencias en que son usadas.

Un importante archivo producido por el analizador es D1. Representa la interpretación o traducción del programa del usuario en tablas de parámetros constantes y variables tal y cual son requeridas por EXECUTOR.

D2 es un archivo también generado por el analizador, que reúne el diccionario de variables, todos los mensajes de error, nombres de variables implicadas en los diferentes errores y otro tipo de especificaciones que se analizarán con profundidad en el capítulo siguiente.

c) EXECUTOR: Una vez que el analizador ha preparado todas las especificaciones para la consistencia de los datos del usuario, EXECUTOR se encarga de la ejecución de las reglas de consistencia impartidas. Para ello necesita como input D1 y los datos básicos que se quiere verificar y corregir (T1).

Este programa genera a su vez dos archivos diferentes (T2 y T3) de los cuales el primero contiene datos en formato binario o punto flotante, que representa el archivo de salida requerido por el usuario en el caso

de que no se detecte ningún error en la fase de ejecución; el segundo (T3) contiene la identificación de los errores detectados en tiempo de ejecución seguidos del cuestionario que contiene esos errores.

d) LISTER: Una vez que se ha terminado con la labor de aplicar las reglas de consistencia establecidas por el usuario a todos los datos, LISTER se encargará de sacar una estadística de errores que permita formarse una idea clara de la calidad de los datos de entrada, cuánto y cómo se corrigió y, en el caso de que se desee hacer una corrección manual, proveerá al usuario de la información necesaria para que en forma expedita identifique cuestionarios y variables con problemas. Para llevar a cabo esta tarea, LISTER hace uso de los archivos D2 y T3 generados en las etapas de compilación y ejecución respectivamente. El primero, como ya se explicó, contiene el diccionario de variables (permite ubicar en el cuestionario de entrada cada una de las variables) y el Diccionario de errores (que provee la descripción de todos los errores posibles y los nombres de las variables involucradas en cada uno de ellos). El segundo archivo (T3) revela los errores que se detectaron, identificados por un número que servirá a LISTER para obtener del diccionario de errores, la descripción respectiva.

e) CORRECTOR: Esta fase del subsistema solamente se ejecutará cuando exista corrección manual de los datos. Para realizar la etapa de corrección, CORRECTOR hará uso del archivo T3, que contiene todos los cuestionarios con error; D2, que contiene el diccionario de variables, y un conjunto de tarjetas preparadas por el usuario, en el que se indican la o las correcciones que se desea efectuar.

Como se puede deducir de la figura 1, CORRECTOR producirá además de un listado de las tarjetas de corrección y diagnóstico de errores de sintaxis, un archivo (T4) que contendrá los cuestionarios corregidos.

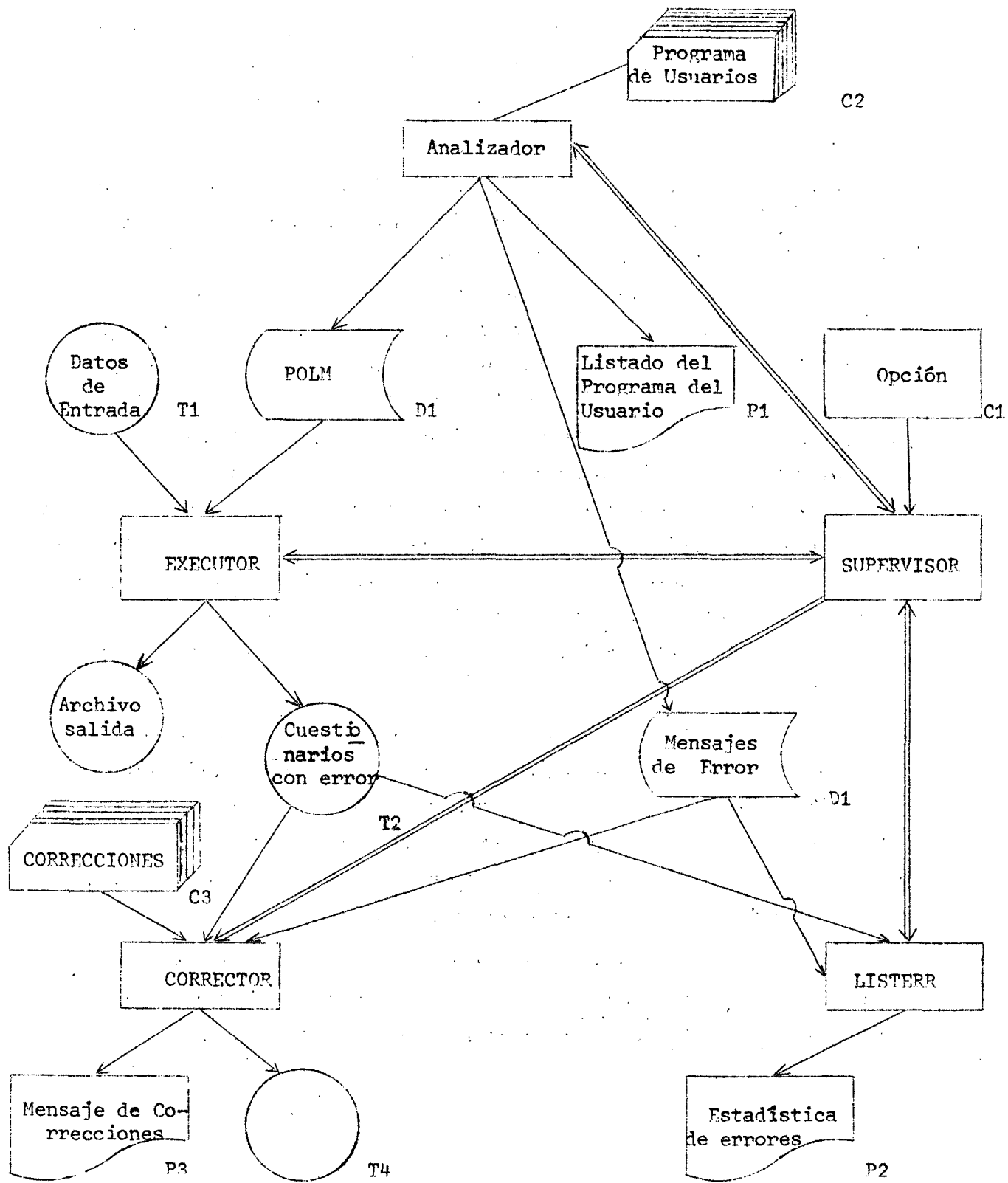


Figura 1

CAPITULO II

Descripción del Lenguaje

A. Restricciones del archivo de entrada

Antes de empezar la descripción y sintaxis del lenguaje propiamente tal, es interesante destacar que CONCOR puede aceptar archivos de entrada compuestos de distintos tipos de registros a condición de que ellos tengan las siguientes características:

- a) Los registros, aunque sean de diferentes tipos, deben ser de longitud fija.
- b) Si hay más de un tipo de registro por caso, cada uno de ellos debe disponer de dos campos que identifiquen al cuestionario (caso) y al tipo de registro respectivamente. Dichos campos deberán ocupar una posición y deben tener un largo fijo en cada uno de los diferentes tipos de registros.

B. Tipos de sentencias

A través de la exposición de las instrucciones del lenguaje, se encontrarán dos tipos de sentencias:

- a) Sentencias no ejecutables, que pueden ser definidas como instrucciones al analizador sintáctico y que por lo tanto no originan ninguna operación en tiempo de ejecución.
- b) Sentencias ejecutables que son todas aquéllas que contienen las normas para la consistencia y corrección de los datos.

Estas sentencias a su vez pueden ser clasificadas en instrucciones de:

- i) verificación
- ii) recodificación y conversión de datos
- iii) aritméticas
- iv) control de flujo del proceso
- v) asignación dinámica
- vi) generación de registros de salida.

C. Codificación de las sentencias

Las sentencias se podrán escribir en hojas de codificación corriente de acuerdo con las siguientes reglas:

<u>Columnas</u>	<u>Descripción del contenido</u>
1-4	Rótulo de la sentencia (opcional)
5	Blanco
6-10	Palabra clave de la operación
11	Blanco
12-71	Operandos
72	Continuación
73-80	Secuencia (opcional)

El rótulo que se especifica en las columnas 1-4 sirve para referirse a una sentencia específica en una bifurcación y está constituido por una combinación de 1 a 4 caracteres alfanuméricos que deben comenzar con un carácter alfabético en columna 1.

Ejemplos:

<u>Válidos</u>	<u>Inválidos</u>
CO01	1001
CONT	CO T
SUBR	SUB+
SEX1	SEX=

Si los operandos de una sentencia exceden la columna 71, se puede continuar su codificación en la columna 12 de la tarjeta siguiente, a condición de que el último carácter sea una coma (,).

Se podrán incluir tantas tarjetas de comentarios como se quiera, codificando un asterisco en la columna 1.

D. Uso de Constantes

En la codificación del lenguaje se aceptan dos tipos de constantes:

- a) Constantes literales: son cuerdas de caracteres válidos en el sistema IBM/370 o 360, formada por 1 a 4 caracteres alfanuméricos encerrados entre apóstrofes. Por ejemplo: '-'; 'ABCD'; '-1'; etc.
- b) Constantes numéricas: son números enteros positivos escritos sin punto decimal, y su máximo valor no puede exceder el valor 2^{30} .
Por ejemplo: 1; 514; 8492; etc.

E. Uso de variables

El lenguaje permite usar variables para referirse a datos en forma simbólica.

Los nombres de variables son cuerdas de hasta cinco caracteres alfanuméricos, el primero de los cuales debe ser alfabético.

Atendiendo a su función, existen dos tipos de variables: aquéllas que sirven para identificar campos del registro de entrada, que se denominarán variables de entrada, y las que identifican resultados de operaciones aritméticas o de recodificaciones, que se llamarán variables creadas.

F. Uso del asterisco (*) como valor de asignación

El lenguaje proporciona la posibilidad de "marcar" ciertas variables con un valor especial (imposible de alcanzar por otros medios) mediante el uso del símbolo asterisco. Este valor puede ser asignado a una variable mediante diversas operaciones del lenguaje teniendo como objetivo poder identificar aquéllas que deberán ser sometidas a algún procedimiento de corrección automático. Las instrucciones que permiten la asignación del valor * son DICT, INIT, SET, RANGE, XRC y DRC, que más adelante se describen.

G. Instrucciones del lenguaje

En la descripción de las instrucciones del lenguaje se especificará en forma individual cuando sean del tipo no ejecutables y deberá suponerse que las demás sí lo son.

1. Instrucciones para definir archivo de entrada e inicializar variablesDICT

Esta instrucción no es ejecutable e indica el comienzo de la definición del diccionario de variables de entrada. Toda tarjeta de descripción de variables debe llevar en blanco las columnas 1 a 11 a fin de diferenciarlas de las sentencias normales de operación. Cada descripción de variable ocupará una tarjeta de acuerdo con el siguiente formato:

<u>Columnas</u>	<u>Descripción del campo</u>
12-16	Nombre de la variable
20-21	Tipo de registro en que se ubica la variable
23-25	Posición inicial (1 ^a columna del campo) de la variable en el registro. Debe ser un número sin signo ajustado a la derecha del campo.
27	Largo en caracteres de la variable
29	Tipo de variable A: variable alfanumérica B: variable binaria N: variable numérica en código EBCDIC
31-50	Se usan sólo en caso de que la variable sea numérica, para definir valores que se desean imputar si el contenido de la variable no es numérico. 31-35 si contiene zona(s) 11(-) 36-40 si contiene zona(s) 12(ε) 41-45 si contiene blanco(s) 46-50 cualquier otro contenido no numérico.

Todo código o valor de imputación debe codificarse ajustado a la derecha del campo respectivo.

Consideraciones en relación al diccionario de entrada

- a) Si el archivo de entrada se compone de un solo tipo de registro, se deben dejar las columnas 20 y 21 en blanco.
- b) Si existe más de un tipo de registro, y hay variables que se repiten en cada uno de ellos y en la misma ubicación, se deberán definir primero dejando las columnas 20 y 21 en blanco.
- c) Si se desea hacer algún tipo de corrección manual, la variable "número del cuestionario" o de cédula deberá ser definida en la misma tarjeta DICT siguiendo las reglas especificadas. Como nombre de esta variable se deberá colocar la palabra clave %NC.
- d) Si hay más de un tipo de registro, la variable que indica el "tipo de registro" deberá ser definida con la palabra clave %CT.
- e) Los valores o códigos de imputación son opcionales y en caso de que el usuario no los especifique, y se detecte un valor no numérico, el sistema le asignará valores estándar de acuerdo con la siguiente relación: $V=10^n$ en que n es el número de dígitos especificado en la columna 27, y V el valor que se imputará.
- f) La definición de variables debe hacerse en estricto orden de ubicación de ellas dentro de cada registro y, en lo posible, teniendo en cuenta el orden de los registros. Excepción de esta regla son las variables que se repiten en cada tipo de registro.
- g) Las variables se definirán en la memoria del computador en 4 bytes cada una, en forma contigua y en el mismo orden en que aparecen en el diccionario.
- h) Las variables de tipo alfanuméricas (A) podrán tener un largo entre 1 y 4 caracteres.

* EJEMPLO DE DEFINICION DE DICCIONARIO DE ENTRADA

DICT	%NC	77	4	N
	%CT	4	2	N

TARJETA # 1

PROV	01	1	1	N	
MUN	01	2	1	N	
ZONA	01	3	1	N	
RESE	01	16	1	N	
HOGAR	01	17	1	N	
TOTHO	01	18	1	N	
DIAEN	01	19	2	N	
MESEN	01	21	2	N	
ANOEN	01	23	2	N	
PAR	01	30	1	N	
RESH	01	33	1	N	
SEXO	01	34	1	N	
ORF	01	35	1	N	
EDAD	01	36	2	N	
NIVIN	01	38	2	N	100
ESTCV	01	40	1	N	
FECUN	01	42	2	N	100
HFALL	01	44	2	N	100

*

TARJETA # 2

P201	02	6	1	N	
P202	02	7	1	N	100
P204A	02	8	1	N	100
P205A	02	11	1	N	100
P206A	02	14	1	N	100
P207A	02	17	1	N	100
P204B	02	9	1	N	100
P205B	02	12	1	N	100
P206B	02	15	1	N	100
P207B	02	18	1	N	100
P308	02	8	1	N	100
P309	02	11	1	N	100
P310	02	14	1	N	100
P311	02	17	1	N	100

En este ejemplo se puede apreciar que:

a) El campo que identifica al número del cuestionario se encuentra en las posiciones 77-80 del registro, tanto en la clase de tarjeta 1 como 2. El analizador entenderá que este campo es el que indica el número del cuestionario por la palabra clave %NC.

b) El campo que identifica a la clase de tarjeta es común a ambos y se encuentra en las posiciones 4-5.

El analizador reconocerá que este campo es el que indica la clase de tarjeta por la palabra clave %CT.

Nótese que tanto %NC como %CT deben ser campos comunes a todos los tipos de registros, y que por tanto las columnas 20-21 están en blanco.

c) Nótese que el orden en que se definen las variables de entrada no es necesariamente el mismo en que ellas están dispuestas en el registro. En este ejemplo, se definió primero la variable P205A y posteriormente la P204B pese a que la primera se encuentra en la posición 11 y la segunda en la posición 9.

d) También debe tenerse en cuenta que un mismo campo puede ser asociado con más de una variable como es el caso de las variables P204A y P308; P205A y P039 etc.

FILT

Esta es una instrucción no ejecutable que debe estar codificada si y sólo si el archivo de entrada está compuesto por más de un tipo de registro. Tiene por objeto indicar la dirección en el programa donde se quiere comenzar el análisis de cada uno de los tipos de registros que se pueden encontrar.

Su formato es:

FILT (K1,L1,K2,L2,...,Kn,Ln) ,LE

en que:

Ki: es una constante igual a la usada para definir el tipo de registro Ki.

Li: es un rótulo que indica la dirección donde se quiere empezar con el tratamiento del tipo de registro Ki.

LE: es un rótulo que indica la dirección donde se bifurca si viene algún tipo de registro no contemplado.

* EJEMPLO DE LA SENTENCIA FILT
FILT (1,T1,2,T2),ERR

En este ejemplo se le indica al analizador que si detecta la clase de tarjeta 1 debe bifurcar a la sentencia que tiene el rótulo T1. En cambio, si detecta la clase de tarjeta 2, bifurque a T2.

También se le indica que si detecta cualquier otro tipo de tarjeta, bifurque a la instrucción que tiene como rótulo ERR.

INIT

Es una instrucción no ejecutable que indica al compilador que un grupo determinado de variables debe tener determinados valores al iniciarse la ejecución.

Su formato es:

INIT	$v_1=k_1$, $v_2=k_2$...
------	-----------	-------------	-----

en que:

v_i : es el nombre de una variable de tipo numérica.

k_i : es una constante numérica cuyo valor será cargado en la variable v_i .

En particular este valor puede ser *.

* EJEMPLO DE UNA SENTENCIA INIT

INIT SEXO=1,NHV=0,ZONA=1

En este ejemplo se le ha indicado al analizador que la variable SEXO deberá ser definida por 1, la variable NHV por 0, y la variable ZONA por 1. Estos valores deberán ser considerado como iniciales en el momento en que se comienza a ejecutar la consistencia y el usuario podrá alterar posteriormente los contenidos de las variables según lo necesite.

2. Instrucciones para el manejo del flujo de ejecución

Este tipo de instrucciones tiene como objetivo alterar el flujo secuencial de ejecución ya sea condicional o incondicionalmente.

GOTO

Esta instrucción provoca una bifurcación incondicional a una sentencia diferente de la que sigue inmediatamente en secuencia.

Su formato es:

GOTO	L
------	---

en que:

L: es un rótulo que indica la instrucción en la cual se desea continuar la ejecución. La instrucción que está a continuación de un GOTO debe llevar obligatoriamente un rótulo.

IF

Si el valor lógico de una relación entre dos variables o dos constantes es verdadero, el control se transfiere a una sentencia ejecutable cuyo rótulo se indica. Si el valor lógico de dicha relación es falso, la operación continúa con la siguiente sentencia en secuencia.

IF	(v_1 .op-rel. v_2)	GOTO r
----	--------------------------	--------

en que:

v_1, v_2 : deben ser variables o constantes del mismo tipo (si v_1 es variable o constante literal, v_2 debe ser literal de igual largo; si v_1 es variable o constante numérica, v_2 debe ser también numérica).

op-rel: si v_1 y v_2 son literales, se permite el uso de los operadores de relación EQ (igual) o NE (distinto). Si, en cambio, v_1 y v_2 son numéricas, se permite además el uso de los operadores LT (menor que), GT (mayor que), LE (menor o igual a) y GE (mayor o igual a).

r: es cualquier rótulo de una sentencia ejecutable del programa, a excepción de un rótulo de sentencia PROC.

* EJEMPLO DE LA SENTENCIA IF

```
IF (A,LT,1) GOTO M1
IF (B,GT,5) GOTO M5
```

La primera sentencia de este ejemplo ordena bifurcar la sentencia que tiene como label M1 si A es menor que 1. En caso de que A no sea menor que 1, la ejecución continúa con la sentencia que le sigue en secuencia. La segunda sentencia ordena bifurcar a M5 si B es mayor que 5. En caso de que B sea menor o igual que 5, la ejecución prosigue con la próxima sentencia.

STOP

Indica que no se efectuarán más operaciones para el registro de entrada en proceso. Luego de alcanzada la sentencia STOP, CONCOR lee un nuevo registro de entrada y reinicia el proceso. Toda sentencia que aparezca a continuación de STOP debe llevar un rótulo.

STOP

Opcionalmente, STOP puede llevar rótulo.

3. Instrucciones de recodificación

En ocasiones, el usuario quiere definir una variable como una recodificación o transformación de otra previamente definida. Si se quiere efectuar una operación de este tipo, el usuario puede elegir tres formas posibles:

XRC

Recodifica una variable numérica cualquiera "ve" en otra "vs" por medio de parámetros posicionales que indican el valor que tomará la variable "vs" para cada uno de los posibles valores de "ve".

XRC vs=ve,k₀, [k₁,k₂,...,]k_n

en que:

vs: variable resultado de la recodificación. Debe ser numérica, y puede estar previamente definida o ser definida en esta sentencia.

ve: variable base de la recodificación. Debe ser numérica y estar previamente definida.

k₀: valor que tomará vs si ve tiene el valor 0.

k₁: valor que tomará vs si ve tiene el valor 1.

k_n: valor que tomará vs si ve tiene el valor n.

NOTA: La sentencia XRC admite como parámetro de recodificación el "n".

Ejemplo de una recodificación XRC.

Supongamos que la variable estado civil (ESTC) se quiere recodificar en otra que llamaremos ESTCR de acuerdo con la siguiente tabla de transformación:

<u>ESTC</u>	<u>ESTCR</u>	
1	1	Soltero
2	2	Casado por lo civil
3	2	Casado por la Iglesia
4	2	Conviviente
5	3	Separado legalmente
6	3	Divorciado
7	3	Viudo
8	9	Ignorado

Esta operación se puede efectuar fácilmente por medio de la siguiente sentencia:

```
XRC ESTCR=ESTC,9,1,2,2,2,3,3,3,9
```

Como se puede apreciar, el primer 9 que aparece en la sentencia corresponde al valor que tomará ESTCR si ESTC tiene el valor 0. Puesto que 0 es un código no definido para el estado civil, ESTCR se define como ignorado.

SRC

Es una sentencia que tiene por objeto recodificar una variable numérica cualquiera previamente definida, "ve", en otra también numérica definida o no "vs". La instrucción tiene el siguiente formato:

SRC	vs=ve,k ₀ ,l ₀ ,l ₁ ,...,l _n
-----	--

en que:

ve: variable base de la recodificación

vs: variable resultado de la recodificación

k₀: valor a asignarse a "vs" si "ve" es menor o igual que l₀.

l₀,l₁,...,l_n: límites superiores de intervalos en que se quiere recodificar "ve".

Después de la recodificación, "vs" puede tomar los siguientes valores:

$vs=k_0$	si $ve \leq l_0$
$vs=k_0+1$	si $l_0 < ve \leq l_1$
$vs=k_0+2$	si $l_1 < ve \leq l_2$
.	.
.	.
.	.
$vs=k_0+n$	si $l_{n-1} < ve$

* EJEMPLO DE UNA RECODIFICACION SRC
 SRC EDADR=EDAD,1,4,9,14,19,24,29,34,39,44,99

En este ejemplo se muestra como recodificar la variable edad en grupos quinquenales de acuerdo con la siguiente tabla de transformación:

<u>EDAD</u>	<u>EDADR</u>
0-4	1
5-9	2
10-14	3
15-19	4
20-24	5
25-29	6
30-34	7
35-39	8
40-44	9
45 y +	10

DRC

Recodifica una variable numérica previamente definida, "ve", en otra también numérica definida o no "vs". El formato de la instrucción es el siguiente:

DRC vs=ve,l₁,k₁,l₂,k₂,...,l_n,k_n

en que:

ve: variable base de la recodificación

vs: variable resultado de la recodificación

l₁,k₂,...,k_n: valores que toma "vs" dependiendo del valor que tenga "ve" (puede ser *)

Entonces:

vs=k ₁	si	ve ≤ l ₁
vs=k ₂	si	l ₁ < ve ≤ l ₂
.	.	.
.	.	.
.	.	.
vs=k	si	l _{n-1} < ve

* EJEMPLO DE UNA RECODIFICACION DRC
 DRC NIVR=NIV,0,1,10,7,15,2,16,3,20,7,25,4,
 26,5,30,7,38,6,99,7

En este ejemplo se ha recodificado el nivel de instrucción (NIV) en la variable NIVR de acuerdo a la siguiente tabla de transformación:

<u>NIV</u>	<u>NIVR</u>	
0	1	Sin instrucción
1-10	7	Ignorado
11-15	2	Primaria incompleta
16	3	Primaria completa
17-20	7	Ignorado
21-25	4	Secundaria incompleta
26	5	Secundaria completa
27-30	7	Ignorado
31-38	6	Universitaria y superior
39-99	7	Ignorado

4. Instrucciones de Cálculo (COMPUTE)CPT

Esta instrucción permite definir una variable como el resultado de evaluar una expresión aritmética tan simple o tan compleja como se quiera. En su forma más simple, la definición puede consistir en un traspaso de valor de una variable a otra.

Su formato es:

CPT	v=Exp.
-----	--------

en que:

v: es el nombre de la variable que se quiere definir

Exp: es una expresión aritmética. Para la codificación de una expresión aritmética se usarán los siguientes símbolos como operadores:

+ para la adición

- para la substracción

* para la multiplicación

/ para la división

Adicionalmente, se puede hacer uso de los paréntesis () para romper la prioridad de los operadores. La expresión se evaluará considerando que los símbolos * y / tienen ambos igual prioridad pero mayor que los otros dos. A su vez, los operadores + y - tienen también igual prioridad.

Entre operadores que tienen igual prioridad, la expresión se evalúa de izquierda a derecha.

Si se desea romper la prioridad intrínseca de los operadores, se puede hacer uso de los paréntesis al igual que en el álgebra.

* EJEMPLO DE LA INSTRUCCION CPT:

CI CPT $X=EDAD/5+1$
 C2 CPT $Y=(A+B)/(A-B)*4$
 C3 CPT $Z=(A+B)/((A-B)*4)$

En este ejemplo se muestran tres instrucciones CPT que analizaremos en forma individual:

C1: Define la variable X como $X=\frac{EDAD}{5}+1$. Un punto importante de destacar es que la división se efectúa entre números enteros y el resultado es otro entero. Los decimales son truncados sin efectuar ningún redondeo. Teniendo en cuenta esto, se pueda apreciar que X queda definida de la siguiente forma:

EDAD	X
0-4	1
5-9	2
10-14	3
15-19	4
20-24	5
95-99	19

C2: Define la variable Y como $Y=4 \cdot \frac{A+B}{A-B}$. Puesto que la división y la multiplicación tienen ambas la misma prioridad, la expresión se evalúa de izquierda a derecha efectuándose por tanto primero la división. De aquí que el resultado de la expresión sea el ya especificado.

C3: Define la variable Z como $Z=\frac{A+B}{4(A-B)}$

Siendo una expresión igual a la anterior, la introducción de los paréntesis rompe la prioridad de los operadores, causando que $((A-B)*4)$ forme una sola subexpresión.

Instrucciones de Verificación

El lenguaje dispone de dos instrucciones para verificar la consistencia de datos. Una resuelve el problema de la consistencia de rangos y la otra la consistencia entre variables.

Además, existe una instrucción que verifica o inspecciona si el contenido de una o más variables contiene(n) el valor "*", relacionando dicho valor con un error detectado previamente y que es necesario corregir.

RANGE

Verifica que el contenido de una o más variables (valor o código) esté en el o los intervalos definidos en la misma instrucción. Si se detecta un error, la instrucción imputará el valor que se especifica al final de ella. En particular, este valor puede ser "*".

Su formato es:

$\text{RANGE } v_1 \left[\begin{array}{c} \text{TO} \\ v_2 \end{array} \right], \dots, v_n, \dots, v_n = (L_1 \left[\begin{array}{c} - \\ L_2 \end{array} \right], L_n) \text{ ELSE } K$
--

en que:

v_1, v_2, \dots, v_n : son nombres de variables de entrada

L_1, L_2, \dots, L_n : son límites inferiores y superiores de intervalos válidos de las variables especificadas.

K: valor que se imputará a la(s) variable(s) en caso de que se detecte un error de rango.

Consideraciones en relación a esta instrucción

a) Para que se especifique más de una variable en una instrucción RANGE, es necesario que todas ellas tengan el mismo campo de variación o intervalos válidos.

Si hay una serie de variables de entrada consecutivas que tienen el mismo intervalo válido, no es necesario especificar cada uno de sus nombres. Basta especificar el primero seguido de la palabra clave TO y el último nombre de variable que tiene el mismo intervalo.

b) Un punto importante que merece una atención especial por la facilidad que representa para el usuario es la forma como se acusa el error de rango.

En la etapa de compilación, se asocia a cada variable que va a ser sometida a un test de rango, un número correlativo de error de acuerdo con el orden en que aparecen. Dicho número va del 9001 en adelante. Al mismo tiempo el compilador genera en forma automática en el archivo que contiene el diccionario de error, el siguiente mensaje:

V	OUT	OF	RANGE
---	-----	----	-------

en que:

V: es el nombre respectivo de la variable.

En la etapa de ejecución, toda vez que se necesite imputar a la variable que se está verificando el valor de asignación por error (K), se generará un registro en forma automática en el archivo de "errores detectados", el que solamente contendrá el número del cuestionario y el número del error. Posteriormente, LISTER asociará esta información con el diccionario de error para indicar al usuario el detalle del error detectado.

* EJEMPLOS DE LA INSTRUCCION RANGE

```
R1 RANGE SEXO,ASES,HNV = ( 1-2 ) ELSE *
R2 RANGE XI TO X9 = (1-5,8-9,100) ELSE 9
```

En la sentencia R1 se ha especificado que las variables SEXO, ASES y HNV solamente pueden tener los códigos 1 0 2. Cualquier otro valor que ellas tengan será un error, "marcándose" con el valor correspondiente a *.

En la sentencia R2 se puede apreciar el uso de la palabra clave TO. Supongamos que entre X1 y X9 fueron definidas en el diccionario de entrada X2,X3,...,X7,X8. Entonces esta sentencia indica que estas 9 variables tienen solamente como intervalos válidos los valores 1,2,...,5; 8,9 y 100. Cualquier otro valor que ellas tomen será considerado como un error, imputándoseles el código 9.

IFL

Esta instrucción tiene por objeto verificar la consistencia entre variables que se encuentran ligadas por relaciones lógicas.

Su formato es:

IFL	(A)	THEN	(B)	ELSE	C
-----	-----	------	-----	------	---

en que:

- A: es una expresión lógica tan simple o compleja como se quiera.
- B: es otra expresión lógica del mismo tipo que A. Si la expresión lógica A se cumple (TRUE) la expresión B deberá cumplirse.
- C: representa o simboliza una de dos instrucciones posibles, que se ejecutarán en caso de que A se cumpla y B no se cumpla.

Estas dos instrucciones posibles son:

- GOTO L ya vista con anterioridad, y
- WRT K en que K es una constante.

Esta última instrucción generará un registro de error que contendrá solamente:

- El número del cuestionario en que se detectó el error de consistencia, y
 - La constante K que identificará el error detectado.
- K debe ser una constante que sea mayor que cero y menor que 9000.
- $$0 < K < 9000$$

Algunas consideraciones en relación al IF lógico

- a) Una expresión lógica está compuesta por una o más relaciones aritméticas entre una variable y una constante u otra variable, unidas por un operador lógico .AND. u .OR.
- b) Los símbolos que identifican a los operadores en las relaciones aritméticas son los mismos que se vieron en la instrucción IF, excepto que se agregan los símbolos <., >. e .=.

- c) En la evaluación de una expresión lógica se debe tener en cuenta que el operador .AND. tiene prioridad sobre el operador .OR. y que se resuelve de izquierda a derecha. Al igual que en las expresiones aritméticas, se admite el uso de paréntesis para romper la prioridad intrínseca de los operadores.
- d) El operador lógico .AND. representa la intersección de dos conjuntos. En cambio, el operador .OR. representa la unión de ambos conjuntos.
- e) Si hay un conjunto de variables de entrada consecutivas que deben cumplir la misma relación aritmética, se puede hacer uso del operador .TO. en forma análoga a la que se explicó en la instrucción RANGE. El operador lógico .TO. le asociará la misma relación aritmética a todas las variables que comprende, conectando estas relaciones por medio de un operador .AND.

*

EJEMPLO

```
IFL (V001.=.2) THEN (V002.TO.V004.=.BLK) ELSE WRT 10
```

(V002.TO.V004.EQ.BLANK) es equivalente a:

```
(V002.EQ.BLANK.AND.V003.EQ.BLANK.AND.V004.EQ.BLANK)
```

- f) En caso de que dos variables cualesquiera en una expresión deban cumplir idéntica relación aritmética, se podrán unir ambas variables por el operador lógico para después especificar la relación aritmética.

*

EJEMPLO

```
IFL (SEX.EQ.2.AND.LIT.EQ.2) THEN ( A.NE.1 ) ELSE WRT 5
```

*

ES EQUIVALENTE A

```
IFL ( SEX.AND.LIT.-.2 ) THEN (A.NE.1) ELSE WRT 5
```

*

*

OTROS EJEMPLOS DE ESTA INSTRUCCIÓN

```
11 IFL ( A.=.1.AND.B. .2.OR.C.=.3) THEN (Z.=.1) ELSE  
GOTO L11
```

```
12 IFL ( A.=.1.OR.B. .2.AND.C.=.3) THEN (Z.=.1)  
ELSE GOTO L11
```

```
13 IFL ((A.=.1.OR.B. .2).AND.C.=.3) THEN (Z.-.1)  
ELSE GOTO L11
```

Teniendo en cuenta la prioridad de los operadores lógicos, el primer miembro de I1 está compuesto por dos subexpresiones:

A=1 y B>2; C=3

Si cualquiera de estas subexpresiones, o ambas, se cumplen, deberá cumplirse la expresión señalada en el segundo miembro. Es decir, Z=1. Si no se cumple que Z sea igual a 1, se bifurcará a la instrucción que tiene como rótulo L11.

Si no se cumple ninguna de las subexpresiones, no se ejecuta el test especificado en el segundo miembro, prosiguiendo la ejecución con la instrucción inmediatamente en secuencia.

En I2 se puede observar que también hay dos subexpresiones, pero esta vez compuestas por: A=1; B>2 y C=3. Al igual que en I1, si se cumple cualquiera de las dos expresiones, o ambas, deberá cumplirse Z=1.

En I3 se puede observar el uso de los paréntesis para destruir la prioridad intrínseca de los operadores. De esta forma se tienen las siguientes subexpresiones: A=1 o B>2; C=3. Si, y solo si se cumplen ambas subexpresiones, deberá cumplirse A=1.

DIFL

Esta instrucción, al igual que el IFL, tiene por objeto verificar la consistencia entre variables ligadas por relaciones lógicas. Pero, a diferencia de él, que resuelve el problema $A \rightarrow B$, DIFL resuelve el problema $A \leftrightarrow B$. Dicho de otra forma, si A es verdad, B debe ser verdad; pero si B es verdad, también A debe ser verdad.

```
DIFL (A) THEN (B) ELSE C
```

en que:

A y B: son dos expresiones lógicas y

C: simboliza cualquiera de las dos instrucciones siguientes:

- GOTO L

- WRT K

Ejemplo:

En la figura siguiente se puede apreciar un trozo de un cuestionario que ilustra claramente una relación biunívoca (↔). Si la respuesta a la pregunta 233 es 1, las preguntas 234 y 235 deben tener información; a la inversa, si las preguntas 234 y 235 tienen información, la respuesta a la pregunta 233 debe ser 1.

```
* INSTRUCCION DIFL QUE RESOLVERIA EL PROBLEMA DE CON
  SISTENCIA PLANTEADO:
  DIFL (P233.=.1) THEN (P234.AND.P235.NE.BLK) ELSE
    WRT 405
* O LO QUE ES LO MISMO:
  IFL (P233.=.1) THEN (P234.AND.P235.NE.BLK)
    ELSE WRT 405
  IFL (P234.AND.P235.NE.BLK) THEN (P233.=.1)
    ELSE WRT 405
```

<p>233. Está Ud. embarazada actualmente?</p> <p>SI <input type="checkbox"/> 1 NO <input type="checkbox"/> 2 NO SABE <input type="checkbox"/> 3</p> <p>(pase a 236) (pase a 236)</p>	<p><input type="checkbox"/></p> <p>49</p>
<p>234. Para cuando espera el nacim.?</p> <p>_____, 19 _____</p> <p>(mes) (año) No sabe</p> <p>235. Prefiere tener un varón o una niña?</p> <p>Varón <input type="checkbox"/> 1 Niña <input type="checkbox"/> 2 Cualquiera <input type="checkbox"/> 3</p> <p>de los dos</p>	<p><input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/></p> <p>50 52</p> <p><input type="checkbox"/></p> <p>54</p>
<p>236.</p>	

Figura 2: Ilustración de una relación biunívoca.

En el ejemplo ilustrado, se supone que BLK es el nombre de una variable que representa blanco o ausencia de información.

VERIF

Inspecciona el contenido de la o las variables numéricas que aparecen en la sentencia, con el solo objetivo de determinar si su contenido es "*".

Su formato es:

VERIF	(V ₁ , RUT ₁)	[(V ₂ , RUT ₂), ..., (V _n , RUT _n)]
-------	--------------------------------------	---

en que:

V_i: variable a ser inspeccionada.

RUT_i: rutina a ser ejecutada si la variable que se inspecciona (v_i) contiene un "*". Si la variable v_i contiene un "*", se ejecuta la rutina RUT. Una vez ejecutada, se prosigue con la inspección de la siguiente variable v_{i+1} y así sucesivamente hasta terminar con la lista. Si la variable v_i no contiene un "*", se prosigue con la inspección de la siguiente variable en la lista.

Al terminar la inspección de la última variable, se prosigue con la ejecución de la sentencia inmediatamente a continuación de la VERIF.

Rutinas invocadas por VERIF

Las rutinas invocadas por VERIF tienen por objeto someter a un determinado tratamiento una variable que contenga "*".

En realidad, esta rutina puede ser considerada como una subrutina que persigue asignar un valor a una variable en función de otra u otras variables. Por ser una sub-rutina, tiene una entrada y una salida especiales, lo cual permite, una vez ejecutada, continuar con el flujo normal de ejecución. La entrada a esta sub-rutina está identificada por una sentencia cuyo código de operación es PROC, la cual lleva como rótulo el mismo que se ha codificado en la sentencia VERIF(RUT_i).

El formato de esta sentencia es:

ROT	PROC
-----	------

La salida de la sub-rutina sólo puede efectuarse por medio de una sentencia cuyo código de operación es RETURN. Esta sentencia tiene por objeto, por una parte, retornar el control de ejecución al programa principal, ya sea para la inspección de otra variable en la lista de VERIF, o bien para la ejecución de la siguiente sentencia a continuación de ella; por otra, señala al analizador que la sub-rutina PROC ha terminado. Debe ser, por tanto, única en una sub-rutina.

Resumiendo lo anterior, a una sub-rutina PROC sólo se puede llegar mediante VERIF. De ella sólo se puede salir mediante la sentencia RETURN.

```
*          UNA APLICACION DE LA INSTRUCCION VERIF
VERIF (SEXO,SEXE)
STOP
SEXE PROC
      IF      (EDAD.GT.14) GOTO TFEC
      CPT     SEXO = SEXA
      GOTO    FIN
TFEC  IF      (FEC.EQ.BLK) GOTO MASC
      CPT     SEXO = 2
      GOTO    FIN
MASC  CPT     SEXO = 1
FIN   RETURN
```

En este ejemplo se hace uso de la instrucción VERIF para investigar si la variable SEXO ha sido previamente "marcada" con un *, lo cual implicaría que se detectó un error en dicha variable. En caso de haberse marcado, se bifurcará a la sub-rutina que tiene como rótulo SEXE. En la sub-rutina se investiga si la edad del individuo es mayor que 14, en cuyo caso, y de ser mujer, debería contener información en la variable FEC (fecundidad). Si fuese de sexo masculino, la variable FEC debería estar en blanco o lo que es lo mismo, conteniendo el código que asimile la categoría "NO APLICABLE". Por otra parte, si el individuo es menor de 15 años, se imputa en forma aleatoria el sexo, para lo cual el programa principal debe ir redefiniendo la variable SEXA por algún procedimiento que de una buena distribución de la variable SEXO.

6. Instrucciones para Asignación Dinámica

Hay ocasiones en que, detectado un error en una determinada variable, se desea corregir imputándole un valor por medio de la técnica de asignación dinámica o "hot deck".

El lenguaje posibilita el uso de esta técnica por medio de cuatro instrucciones que a continuación se describen:

DMN

Esta sentencia no es ejecutable y solamente permite declarar la dimensión de la o las matrices que se utilizarán en la asignación dinámica

```
DMN M1(L1,C1) ,M2(L2,C2),...,MN(LN,CN)
```

en que:

- M1: nombre de la matriz que se utilizará para la imputación.
- L1: constante numérica que indica el número de líneas de la matriz.
- C1: constante numérica que indica el número de columnas de la matriz.

* EJEMPLO DE LA SENTENCIA DMN

```
DMN ED(20,5),HI(25,20)
```

STORE

Esta sentencia le indica al sistema que a continuación de ella vienen la o las tarjetas que contienen los valores que servirán para inicializar la matriz de asignación.

```
STORE M
```

en que:

- M: es el nombre de la matriz que se quiere inicializar. A continuación de la sentencia STORE deben venir la o las tarjetas que con-

tienen los valores iniciales separados por coma ",". Para indicar el fin de tales valores, basta que el último de ellos tenga un espacio a su derecha. Los valores son almacenados fila por fila, definiendo el primero a M_{11} , el segundo a M_{12} y así sucesivamente. La tarjeta parámetro podrá contener valores codificados entre las columnas 1 y 78 inclusive. Si se desea continuar con otra tarjeta parámetro, basta que el último valor codificado tenga una coma "," a su derecha. En todo caso, una tarjeta parámetro de este tipo se puede, opcionalmente, terminar en cualquier columna.

* EJEMPLO
DMN MAT1(4,5),MAT2(3,3)
STORE MAT1
1,2,3,4,5,
6,7,8,9,10,
11,12,13,14,15,
16,17,18,19,20
STORE MAT2
1,2,3,4,5,6,7,8,9

En el primer caso, se usó una tarjeta parámetro para cada fila.
En el segundo, se utilizó una sola tarjeta para definir la matriz completa.

ASGN

Esta sentencia permite imputar a una variable un valor proveniente de una matriz de asignación.

ASGN	V=M(VL,VC)
------	------------

en que:

V: nombre de la variable a la cual se quiere imputar un cierto valor.

M: nombre de la matriz de asignación.

VL,VC: nombre de las variables que indican fila y columna de la matriz de la cual se extraerá el valor a imputar. Si los valores de las variables fila o columna fuesen menores o iguales que cero, el sistema presumirá que se trata de la primera fila o columna. En cambio,

si ellos fuesen mayores que el número de filas o columnas especificadas en la sentencia DMN, presumirá que corresponden a la última fila o columna.

ACT

Esta sentencia permite actualizar la matriz de asignación y es la que en definitiva permite imputar en una forma probabilística un cierto valor a una variable. Su función es por lo tanto traspasar el valor de una variable a una celda de la matriz de asignación. La celda de la matriz quedará determinada por un par de variables que identificarán la fila y columna en que se encuentra localizada. Este par de variables puede ser el resultado de combinar entre sí tantas variables como se necesiten. Al igual que en la sentencia ASGN, las variables fila y columna no deberán tomar un valor que exceda el especificado en la instrucción DMN, ni tomar un valor menor que 1.

ACT	M(VL,VC)=B
-----	------------

en que:

M: nombre de la matriz de asignación

VL y VC: nombre de las variables fila y columna respectivamente.

V: nombre de la variable de la cual se extrae el valor con que se actualiza la matriz M.

* UN EJEMPLO COMPLETO DE ASIGNACION DINAMICA
 * IMPUTACION DEL # DE HIJOS A MUJERES QUE NO DECLARARON LA FECUNDIDAD.

DICT

SEXO	11 1 B
EDAD	12 1 B
CIVIL	13 1 B
NIVEL	17 1 B
HIJOS	25 1 B

DMN MAFEC(24,16)
 STORE MAFEC

0,0,0,0,0,1,1,1,1,1,1,2,3,3,4,1,	00-05 AÑOS DE ESTUDIOS SOLTERAS
0,0,0,0,0,1,1,1,1,1,1,2,3,3,4,1,	6 AÑOS DE ESTUDIOS SOLTERAS
0,0,0,0,0,1,1,1,1,1,1,2,3,3,4,1,	07-11 AÑOS DE ESTUDIOS SOLTERAS
0,0,0,0,0,1,1,1,1,1,1,2,3,3,3,1,	12 AÑOS DE ESTUDIOS SOLTERAS
0,0,0,0,0,1,1,1,1,1,1,2,2,2,2,1,	13 Y +AÑOS DE ESTUDIOS SOLTERAS
0,0,0,0,0,1,1,1,1,1,1,1,3,3,4,1,	IGN. AÑOS DE ESTUDIOS SOLTERAS
0,1,1,1,1,2,2,2,2,2,4,2,5,5,5,4,	00-05 AÑOS DE ESTUDIOS CASADAS Y UNIDAS
0,1,1,1,1,2,2,2,2,2,4,4,5,5,5,4,	6 AÑOS DE ESTUDIOS CASADAS Y UNIDAS
0,1,1,1,1,1,1,1,1,1,3,4,4,4,4,4,	07-11 AÑOS DE ESTUDIOS CASADAS Y UNIDAS
0,1,1,1,1,1,1,1,1,1,2,3,4,5,5,4,	12 AÑOS DE ESTUDIOS CASADAS Y UNIDAS
0,1,1,1,1,1,1,1,1,1,2,3,4,4,4,4,	13 Y +AÑOS DE ESTUDIOS CASADAS Y UNIDAS
0,1,1,1,1,2,2,2,2,2,4,3,5,5,5,4,	IGN. AÑOS DE ESTUDIOS CASADAS Y UNIDAS
0,1,1,1,1,2,2,2,2,2,2,4,4,4,5,4,	00-05 AÑOS DE ESTUDIOS VIUDAS SEP. Y DIV.
0,1,1,1,1,2,2,2,2,2,2,3,4,4,5,4,	6 AÑOS DE ESTUDIOS VIUDAS SEP. Y DIV.
0,1,1,1,1,1,1,1,1,1,2,3,4,4,4,4,	07-11 AÑOS DE ESTUDIOS VIUDAS SEP. Y DIV.
0,1,1,1,1,1,1,1,1,1,1,3,3,4,4,4,	12 AÑOS DE ESTUDIOS VIUDAS SEP. Y DIV.
0,1,1,1,1,1,1,1,1,1,1,2,3,3,4,4,	13 Y +AÑOS DE ESTUDIOS VIUDAS SEP. Y DIV.
0,1,1,1,1,2,2,2,2,2,2,2,4,4,5,4,	IGN. AÑOS DE ESTUDIOS VIUDAS SEP. Y DIV.
0,0,0,0,0,1,1,1,1,1,2,3,3,3,4,3,	00-05 AÑOS DE ESTUDIOS; IGNORADO(EST.CIV.)
0,0,0,0,0,1,1,1,1,1,2,3,3,3,4,3,	6 AÑOS DE ESTUDIOS; IGNORADO(EST.CIV.)
0,0,0,0,0,1,1,1,1,1,2,3,3,3,4,3,	07-11 AÑOS DE ESTUDIOS; IGNORADO(EST.CIV.)
0,0,0,0,0,1,1,1,1,1,1,3,3,3,4,3,	12 AÑOS DE ESTUDIOS; IGNORADO(EST.CIV.)
0,0,0,0,0,1,1,1,1,1,1,2,3,3,4,3,	13 Y +AÑOS DE ESTUDIOS; IGNORADO(EST.CIV.)
0,0,0,0,0,1,1,1,1,1,2,3,3,3,4,3	IGN AÑOS DE ESTUDIOS; IGNORADO(EST.CIV.)

```

IF (SEXO.NF.2) GOTO READ
IF (EDAD.LT.15) GOTO READ
SRC EDADR=EDAD,1,15,16,17,18,19,20,21,22,23,24,29,34,39,44,98,99
SRC ANOSR=NIVEL,1,5,6,11,12,19,20
XRC CIVIR=CIVIL,0,0,1,1,2,2,2,3,3,3
CPT ANOSR = CIVIR*6+ANOSR
IF (HIJOS.NE.99) GOTO ACTU
ASGN HIJOS=MAFEC(ANOSR,EDADR)
GOTO READ

```

ACTU ACT MAFEC(ANOSR,EDADR)=HIJOS

*

*

ACA SE DEBE INSERTAR UNA INSTRUCCION PARA GRABAR EL REGISTRO

READ STOP

7. Transformación de Variables Alfanuméricas a Numéricas

Con frecuencia se presenta el caso en que hay variables que tienen códigos mixtos, es decir, numéricos y alfanuméricos, haciéndose necesario convertir estos últimos a numéricos a fin de permitir un tratamiento normal de estas variables.

TRANS

Transforma una variable de tipo alfanumérica (A) en numérica. Los códigos alfanuméricos son convertidos en numéricos de acuerdo con una tabla de conversión que se especifica en la misma instrucción.

TRANS	VS=VE, L ₁ , K ₁ , ..., L _n , K _n , ERR=K _e
-------	--

en que:

- VS: variable resultado de la conversión (numérica).
- VE: variable que se quiere convertir.
- L_i: constante literal usada para compararse con el contenido de VE. Debe estar entre apóstrofes (') y debe contener tantos caracteres como el largo del campo definido para VE. Si contiene menos caracteres, se rellenará con blancos por la derecha. Si contiene más caracteres, se truncará por la derecha.
- K_i: constante numérica que se imputará a VS si VE tiene el valor L_i.
- K_e: constante numérica que se imputará a VS si VE tiene un valor distinto de los L_i especificados.

Algunas consideraciones en relación a esta sentencia

La sentencia TRANS determina un valor de traducción que se imputa a VS en función del valor que tiene VE de acuerdo con la siguiente regla:

- Se inspecciona si el valor de VE es numérico. En caso de que lo sea, se imputa dicho valor.
- Si el valor de VE no es numérico, se verifica si coincide con alguna de las constantes literales (L_i) especificadas, imputando la constante numérica correspondiente (K_i).

c) Si el valor de VF fuese distinto de los ya mencionados, se imputará la constante que se especifica como K_e .

* EJEMPLO DE UNA INSTRUCCION TRANS
 TRANS EDADR,EDAD,'-1',101,'-2',102,'-3',103,'--',109,ERR=109

En este ejemplo se presenta la variable edad, la cual puede tomar los valores alfanuméricos que se muestran en la lista. La instrucción verificará como primera medida si el contenido de la variable es numérico, en cuyo caso convertirá directamente su valor a binario. En caso de que no lo sea, entrará a ver si está en la lista de conversión para imputarle el valor de traducción correspondiente. Si no lo encuentra, le imputará a EDADR el valor 109.

8. Definición de Variables

SET

Esta instrucción permite definir tantas variables como se quiera con una constante común para todas ellas. Esta constante debe ser numérica y, en particular, puede ser un asterisco (*).

SET K ON (V_1 [TO V_i], V_n)
--

en que:

K: es la constante numérica con que se definirán las variables.

V_i : son los nombres de las variables que se quieren definir.

* EJEMPLO DE UNA INSTRUCCION SET
 SET 0 ON (Q101 TO Q108,Q310,Q401 TO Q408)

Con esta instrucción se definen las variables especificadas en la lista con el valor 0. Nótese que en ella se puede hacer uso de la palabra clave TO a fin de facilitar la labor de definición en caso de que se necesite definir un gran número de variables.

9. Instrucciones de Salida

El lenguaje cuenta con tres sentencias que provocan, a opción del usuario, generación de registros en un archivo de salida. Dos de ellas generan en realidad el archivo "resultado" y la restante el archivo con los errores detectados en el proceso de consistencia.

WRT

Genera un registro de error que se compone de una constante que indica el tipo de error y el valor de la o las variables involucradas en una inconsistencia. Adicionalmente, el sistema le anexa el número del cuestionario si éste ha sido especificado en la sentencia DICT.

WRT	K [V ₁ , ..., V ₆]
-----	---

en que:

K: es la constante que identifica el tipo de error y

V_i: es el nombre de la(s) variable(s) involucrada(s) en la inconsistencia.

Si el usuario así lo desea, puede omitir las variables y, en caso de ser variables de entrada, LISTERR obtendrá los valores desde el cuestionario. Si se omiten las variables y LISTERR no encuentra las variables en el diccionario de entrada, pondrá asteriscos como valores de ellas. En el caso particular de la instrucción WRT como opción de la instrucción IFL, debe especificarse sólo la constante K y por tanto omitirse el nombre de las variables.

*

UN EJEMPLO DE LA INSTRUCCION WRT
WRT 400,EDAD,NHNV

En este ejemplo se generará un registro de error que contendrá los siguientes datos:

- a) La constante 400, que servirá para relacionar este registro con la descripción del error que se verá con posterioridad en este manual (DICTE).
- b) El número del cuestionario, el que permitirá a LISTERR poder ubicar el documento con error.
- c) Los valores de las variables EDAD y NHMV en conflicto.

PUT

Ocasiona que se grabe un registro en el archivo de salida. El registro se compone de los campos definidos por las variables cuyos nombres se citan en el texto de la sentencia.

PUT	$V_1(l_1)$	$[, V_2(l_2) \dots]$
-----	------------	----------------------

en que:

V_i : variable previamente definida en el programa.

l_i : largo del campo en el registro de salida. Toda variable numérica tiene formato binario de una palabra, y el usuario puede desear que el respectivo campo del registro de salida sea de menor longitud; en este caso, l_i puede tomar valores 3, 2 ó 1, según sea el largo deseado. El campo de salida será obtenido tomando los últimos 3, 2 ó 1 bytes de la variable designada.

La aparición de una sentencia PUT ocasiona la emisión de un mapa descriptivo en el informe del proceso de traducción. Ese informe consta de: el nombre de la variable, su posición inicial en el registro y su longitud.

Es necesario que todos los registros descritos en distintas operaciones PUT tengan a lo menos un campo común, que permita diferenciarlos dentro del archivo de salida.

* EJEMPLO DE UNA INSTRUCCION PUT
 PUT DAM(1),TR(1),SEXO(1),EDAD(1),EC(1),
 NIV(1),ASESC(1),TAC(1),CAT(1),OCUP(2),
 RAMA(2),HNV(1),HS(1),HNUA(1),HMUA(1)

Esta instrucción generará un registro de salida de una longitud de 17 bytes en binario.

PUTF

Al igual que la sentencia PUT, ocasiona que se grabe un registro en el archivo de salida. A diferencia del anterior, se obtiene la salida en formato punto flotante con un largo fijo de 4 bytes por variable.

PUTF V_1 [V_2, V_3, \dots, V_n]

en que:

V_i : nombre de variables que se desea grabar en punto flotante en el archivo de salida.

* EJEMPLO
 PUTF DAM,TR,SEXO,EDAD,EC,NIV,ASESC,TAC,CAT
 CCUP,RAMA,HNV,HS,HNUA,HMUA

10. Diccionario de Errores

Si el usuario desea obtener una estadística de errores detectados y eventualmente corregidos, el diccionario de errores deberá estar presente en el programa de consistencia.

DICTE

Es una instrucción no ejecutable que indica el comienzo de la definición del diccionario de errores. No lleva rótulo ni operandos.

DICTE

A continuación de esta sentencia, vienen las tarjetas de definición de tipos de errores con un formato de "parámetros posicionales" definidos a partir de la columna 1 y separados por comas.

K,C,V₁ [V₂,...,V₆] , 'LIT'

en que:

- K: es una constante numérica sin signo, entre 1 y 4 dígitos, menor que 9000. Esta constante es la que permite relacionar los detalles de un error con los errores detectados en tiempo de ejecución.
- C: es un código que indica el tipo de impresión que se desea para el error K. Los códigos posibles son:
- 1: el error no se imprime y sólo se tiene en cuenta para la estadística global de errores.
 - 2: imprime el número del cuestionario, número del error (K), nombre(s) de variable(s) involucrada(s), sus respectivos contenidos y el literal o mensaje del error ('LIT').
 - 3: imprime toda la información que otorga el código 2 más un listado completo del cuestionario.
 - 4: indica que el error corresponde a una variable que se corrige en forma automática. LISTER producirá un listado de los códigos imputados a la variable con sus respectivas frecuencias. En este caso particular, el usuario debe especificar en la instrucción

WRT respectiva el nombre de la variable que se está corrigiendo automáticamente a fin de poder producir el listado de frecuencias.

V_i : son los nombres de las variables involucradas en el error K.
 LIT: es un literal que sirve al usuario para reconocer fácilmente el error de que se trata. Consiste en una cuerda de caracteres alfanuméricos de un largo máximo de 40 posiciones encerradas entre apóstrofes ('').

END

A continuación de la última tarjeta del diccionario de errores debe ir una instrucción END que indica el fin del diccionario y fin del programa. Debe ser, por tanto, la última tarjeta del programa. No debe llevar rótulos ni operandos y es una instrucción no ejecutable.

END

```
* EJEMPLOS DE ESPECIFICACION DE ERRORES.
C1 WRT 400,A1,A2,A3
   GOTO FIN
C2 WRT 401,X1,X2,X3,X4
   GOTO FIN
C3 WRT 402,Z
   GOTO FIN
C4 WRT 403
FIN STOP
   DICTE
400,2,A1,A2,A3,' A2-A3 < A1 '
401,3,X1,X2,X3,X4,' X1+X2+X3+X4 '
402,4,Z,' '
403,2,B,C,' B NO PUEDE SER MAYOR QUE C '
END
```

En los ejemplos se han presentado cuatro aplicaciones diferentes que ilustran el uso de esta instrucción conjuntamente con la sentencia WRT.

En el primer caso, la instrucción C1 hace referencia a la especificación de error 400, generando un registro de error que contendrá los valores de las variables A1, A2 y A3 respectivamente. A su vez la especificación de error identificada por la constante 400 proporciona los nombres de las variables involucradas en la inconsistencia y un comentario que ilustra el error detectado. La constante 2 indica a LISTERR que para este error solamente se desea imprimir el nombre de las variables involucradas en la inconsistencia, sus respectivos valores y el comentario que el usuario ha escrito entre comillas.

En el segundo caso, la instrucción C2 hace referencia a la especificación de error 401, la que es similar a la anterior, diferenciándose solamente en la constante 2, que ha sido cambiada por 3 (segundo parámetro de la especificación de error). La constante 3 le indica a LISTERR que para este tipo de error debe entregar la siguiente información: nombre de las variables involucradas en la inconsistencia con sus respectivos valores; comentario ilustrativo del error; listado de todas las tarjetas o registros de que se compone el cuestionario que contiene el error.

El tercer ejemplo (C3) muestra una aplicación diferente pese a que la instrucción WRT es similar a las anteriores. La especificación del error a que ella hace referencia (402) le especifica a LISTERR (mediante el uso de la constante 4) que la variable Z se está corrigiendo en forma automática y que, por tanto, solamente saque a impresión la distribución de frecuencia de los valores o códigos que se le imputaron. LISTERR proveerá entonces al usuario de la siguiente información:

Variable Z

Códigos imputados	Frecuencia absoluta	Frecuencia relativa
1	25	25.0
2	20	20.0
3	30	30.0
4	25	25.0
Total casos imputados	100	100.0

En el último ejemplo se ve una aplicación en que la instrucción WRT no va acompañada de la lista de variables, haciendo solamente referencia a la especificación de error 403. Pese a ello, LISTERR tomará los nombres de las variables involucradas en la inconsistencia de la especificación de error, y mediante ellos el diccionario de entrada obtendrá la ubicación de cada una de las variables, posibilitando así la impresión de los respectivos valores.

11. Instrucciones al Analizador

A continuación se presentan tres instrucciones al analizador que, por tanto, no generan ninguna instrucción ejecutable.

SPACE

Es una instrucción que tiene como único objetivo manejar el espaciado del listado del programa que produce el analizador.

Su formato es:

SPACE K

En que K es una constante que indica el número de líneas en blanco que se quieren dejar antes de imprimir la próxima instrucción.

```
*          EJEMPLO DE UNA INSTRUCCION SPACE
STOP
SPACE 3
*          CONSISTENCIA DEL REGISTRO DE POBLACION.
SPACE 3
TP RANGE A,B      = (1,2) ELSE *
```

El listado de este grupo de instrucciones producido por el analizador será:

* EJEMPLO DE UNA INSTRUCCION SPACE.

STOP

* CONSISTENCIA DEL REGISTRO DE POBLACION.

TP RANGE A,B = (1,2) ELSE *

EJECT

Al igual que la instrucción SPACE, esta instrucción maneja el espaciado del listado producido por el analizador. Difiere de la anterior en que siempre obliga al analizador a imprimir la próxima instrucción en una nueva página. Si la próxima instrucción cae de por sí en la primera línea de una nueva página, la instrucción EJECT no tiene efecto.

Su formato es simplemente:

EJECT

NOP

Esta es una instrucción que tiene como único objetivo poder dirigir una instrucción o una rutina del programa sin necesidad de darle un rótulo a una instrucción ejecutable.

Su formato es:

ROT NOP

En que ROT es cualquier rótulo válido aceptado por CONCOR.

* EJEMPLO DE LA INSTRUCCION NOP

R1 NOP
IFL (A <. B) THEN (C >. D) ELSE GOTO R2

Este grupo de instrucciones es absolutamente equivalente a:

R1 IFL (A <. B) THEN (C >. D) ELSE GOTO R2

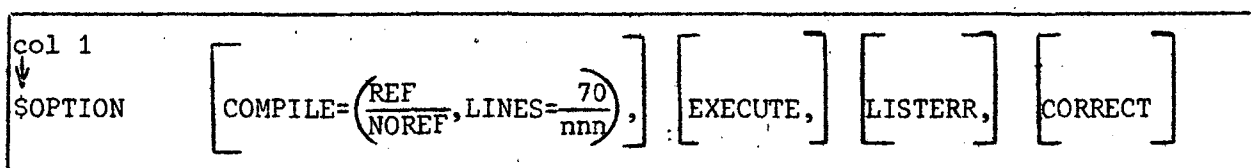
En otras palabras, esta instrucción "No Opera" y sólo sirve para asignar una dirección a un rótulo específico.

12. Instrucción al Monitor

Como se desprende de la figura 1, CONCOR está compuesto por cuatro programas más un monitor que es el que carga en memoria los programas que el usuario desea ejecutar.

La idea de tener un monitor obedece exclusivamente a simplificar al usuario la tarea de especificación de los programas que desea utilizar. Para cumplir con este cometido, se dispone de la instrucción OPTION, que debe ser la primera de todas las sentencias del programa.

Su formato es:



La omisión de alguno de estos parámetros causa la no ejecución de la fase respectiva. Como por ejemplo:

```
$OPTION COMPILE
$OPTION EXECUTE, LISTERR
$OPTION CORRECT
$OPTION COMPILE=(NOREF, LINES=50), EXECUTE, LISTERR
```

En la primera opción se ha especificado que sólo se desea compilar el programa. En otras palabras, solamente se desea ejecutar el analizador sintáctico. Al no especificar los parámetros de la opción COMPILE, el analizador supondrá las estándar; es decir REF y 70 líneas por página.

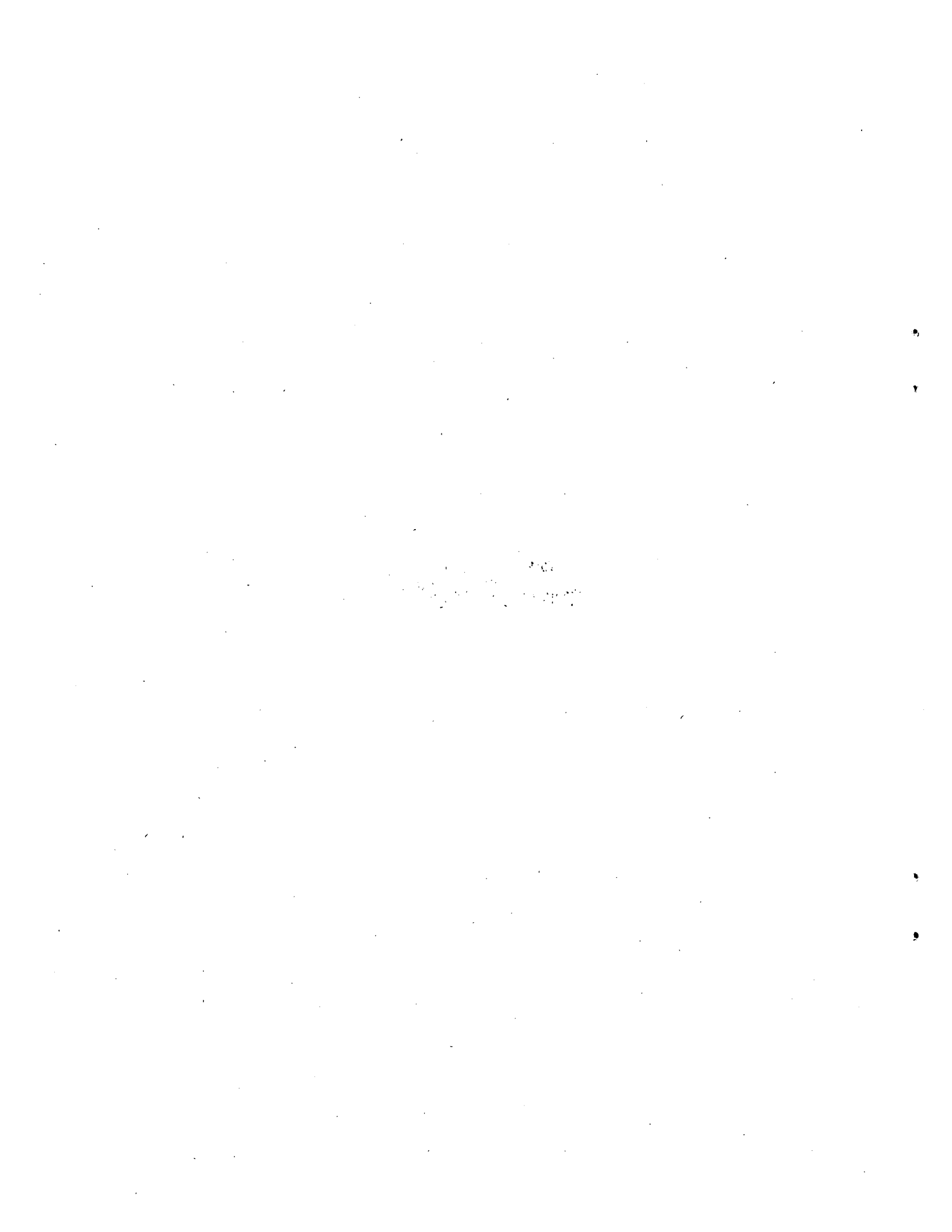
REF le pide al analizador que produzca una lista de referencias de r tulos y nombres de variables, especificando el n mero de la o las sentencias en que se hace menc n de ellos.

En la segunda opci n, s lo se desea la ejecuci n de las reglas de consistencia y el listado de los errores detectados. En este caso, se supone que con anterioridad se ejecut  la opci n COMPILE y por tanto ya se tienen traducidas las reglas de consistencia y de correcci n autom tica.

En la tercera opci n s lo se desea hacer correcciones en el archivo de entrada y por tanto se hace uso del programa CORRECTOR. Este programa, por lo general, se usa solo ya que es necesario analizar primero el output producido por LISTERR, y posteriormente, preparar las correcciones que se desea efectuar en el archivo de entrada.

En la cuarta opci n se ha especificado que se compilar , ejecutar  y listar n los errores detectados. Adem s, se le indica al analizador que no se desea el listado de referencias cruzadas (NOREF) y que en el listado del programa s lo se deben imprimir 50 l neas por p gina.

CAPITULO III
Programa CORRECTOR



A. Explicación general

Pese a que este programa forma parte del sistema, las funciones que desempeña obligan a que se use siempre en forma aislada. Por esta razón se ha preferido presentar las instrucciones a CORRECTOR en forma separada, permitiendo así diferenciarlas claramente del resto del lenguaje. En todo caso, la forma de codificar las instrucciones sigue las reglas previamente descritas.

B. Archivos de input/output y sus restricciones

CORRECTOR trabaja con tres archivos de entrada y uno de salida.

1. Archivo de Entrada

Los tres archivos de entrada para este programa son:

T3: Archivo que contiene todos aquellos cuestionarios en los que se detectó por lo menos un error. Para que CORRECTOR pueda operar, es necesario que este archivo esté ordenado por número de cuestionario y tipo de registro.

El archivo T3 puede ser reemplazado por el archivo T1 (Datos básicos).

D2: Este archivo contiene el diccionario de variables generado por el analizador.

C3: Es un conjunto de tarjetas que contienen las instrucciones para CORRECTOR, en las cuales el usuario da todas las especificaciones que permitirán corregir el archivo T3 o T1.

Las tarjetas deberán estar ordenadas según el número de los cuestionarios que se desea corregir.

2. Archivo de Salida

El único archivo de salida es T4, que teóricamente debería contener cuestionarios sin error.

Si el archivo de entrada que se utilizó fue T3, entonces los cuestionarios contenidos en T4 deberán reemplazar a sus homólogos de T1 para así tener los datos básicos teóricamente limpios. En cambio, si el archivo de entrada utilizado fue T1, el archivo T4 representará de inmediato el archivo de datos básicos limpios.

C. Instrucciones para CORRECTOR

REPLACE

Esta instrucción permitirá reemplazar uno o varios registros de un determinado cuestionario, o bien el cuestionario completo.

Su formato es:

REP %NC=K, $\left\{ \begin{array}{l} \%CT = K_1, [(i_1)] \\ ALL \end{array} \right. \left[, K_2 [(i_2)] , \dots , K_n [(i_n)] \right]$
--

en que:

- K: es el número del cuestionario que se desea reemplazar, o en el cual se desea reemplazar uno o más registros.
- ALL: es la palabra clave que indica que se desea reemplazar todo el cuestionario.
- K_j: indica el tipo de registro que se desea reemplazar.
- i_j: sólo se utiliza cuando un tipo de registro puede estar repetido una o más veces. En este caso, i_j representará el número de orden del tipo de registro K_j que se desea reemplazar.

A continuación de la instrucción REP debe venir la o las tarjetas que se desea reemplazar.

- REP %NC=421,ALL
- * INCLUIR AQUI TARJETAS PARA EL REEMPLAZO
 - * EN ESTE EJEMPLO SE ESTA REEMPLAZANDO TODO EL CUESTIONARIO N°421
- REP %NC=530,%CT=3(2),5
- * INCLUIR AQUI TARJETAS PARA EL REEMPLAZO
 - * EN ESTE EJEMPLO SE HACEN DOS REEMPLAZOS PARA EL CUESTIONARIO N°530
 - a) REEMPLAZA LA SEGUNDA TARJETA 3.
 - b) REEMPLAZA LA TARJETA 5.

DELETE

Esta instrucción permite eliminar uno o más registros de un cuestionario o todo él. En caso de que se haga uso de esta instrucción, el archivo de entrada para CORRECTOR debe ser T1 y no T3.

El formato de esta instrucción es:

$\text{DEL } \%NC=k, \left\{ \begin{array}{l} \%CT=K_1 [(i_1)] \\ \text{ALL} \end{array} \left[,K_2 [(i_2)], \dots, K_n [(i_n)] \right] \right\}$
--

En que K, K_j, i_j y ALL se refieren a los mismos conceptos especificados en la instrucción REPLACE.

- DEL %NC=230,ALL
- * EN ESTE EJEMPLO SE HA BORRADO TODO EL CUESTIONARIO N°230
- DEL %NC=342,%CT=2,3(4),5,6
- * EN ESTE EJEMPLO SE HA BORRADO LAS SIGUIENTES TARJETAS
 - a) LA TARJETA 2
 - b) LA CUARTA TARJETA 3
 - c) LA TARJETA 5 y
 - d) LA TARJETA 6

INSERT

Esta instrucción permite insertar uno o más registros de un cuestionario, o bien un cuestionario completo.

Su formato es:

$\text{INS } \%NC=k, \left\{ \begin{array}{l} \%CT=K_1 [(i_1)] \\ \text{ALL} \end{array} \left[,K_2 [(i_2)], \dots, K_n [(i_n)] \right] \right\}$
--

Al igual que en las dos instrucciones anteriores, K , K_j , i_j y ALL representan los mismos conceptos.

- INS %NC=700,ALL
 * AQUI SE INCLUYE TARJETAS PARA INSERTAR
 * EN ESTE EJEMPLO SE ESTA INTERCALANDO TODO EL CUESTIONARIO N°700
 INS %NC=803,%CT=3(2),3(3),7
 * EN ESTE EJEMPLO SE ESTAN INSERTANDO LAS SIGUIENTES TARJETAS:
- DOS TARJETAS 3, EN SEGUNDO Y TERCER LUGAR RESPECTIVAMENTE
 - UNA TARJETA 7

CHANGE

Esta instrucción se utiliza cuando sólo se quiere cambiar los valores de una o más variables de un cuestionario.

Su formato es:

CHN %NC= K , V_1 [i_1] = K_1 [V_2 [i_2]] = K_2 ,..., V_n [i_n] = K_n]

en que:

- K : es el número del cuestionario que se desea modificar.
 V_j : es el nombre de la variable que se desea corregir.
 K_j : es el valor o código que se desea que asuma la variable V_j .
 i_j : sólo se utiliza en caso de que el tipo de registro en el cual está ubicada la variable V_j pueda estar repetido una o más veces. En este caso, i_j representa el número de orden de ese registro.

- CHN %NC=1543,SEXO=2,EDH(2)=6,EDH(3)=7
 * EN ESTE EJEMPLO SE HAN CAMBIADO LOS VALORES PARA LAS SIGUIENTES VARIABLES:
- A LA VARIABLE SEXO SE LE ASIGNA UN 3
 - A LA VARIABLE EDH EN LA SEGUNDA TARJETA SE LE ASIGNA UN 6
 - A LA VARIABLE EDH EN LA TERCERA TARJETA SE LE ASIGNA UN 7.

4

7

2

2

7



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)