Natural ADABAS

ÍNDICE

Autor: Yorman Guadarrama Martínez

INTRODUCCIÓN

Este documento pretende transmitir de manera clara, práctica y resumida las bondades del producto Natural Adabas bajo entorno Mainframe y está dirigido a programadores que se inician en este lenguaje.

Al finalizar su lectura se alcanzará un conocimiento cualificado que le permitirá realizar desarrollos con cierto nivel de complejidad.

Proporciona las herramientas básicas, intermedias y avanzadas para el desarrollo de aplicaciones, abarcando los temas más importantes, usuales y presentes en el área laboral.

Se apoya de ilustraciones y ejemplos adaptados a cada tema, con el fin hacer un buen uso del gestor de base de datos (ADABAS) a través de Natural.

ADABAS (Adaptable Database System) es un sistema gestor de BD de listas invertidas de alto rendimiento bajo el sistema operativo mainframes de IBM (Z/OS). Natural es el lenguaje usado para acceder a la BD ADABAS creados ambos por la empresa alemana Software AG.

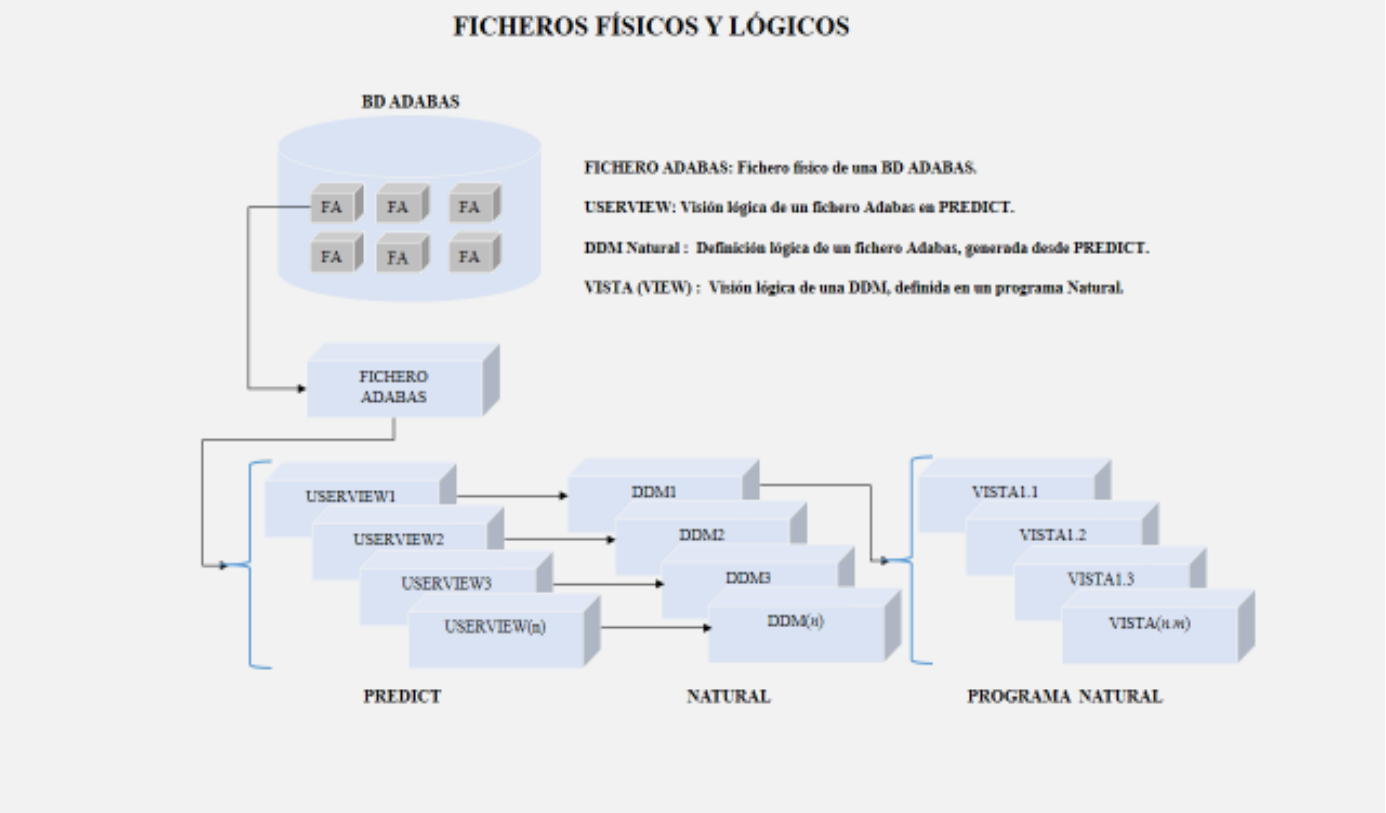
Se basa en un modelo de datos orientado a ficheros. Las entidades se representan en registros que están divididos en campos que representan sus propiedades y están agrupados en ficheros.

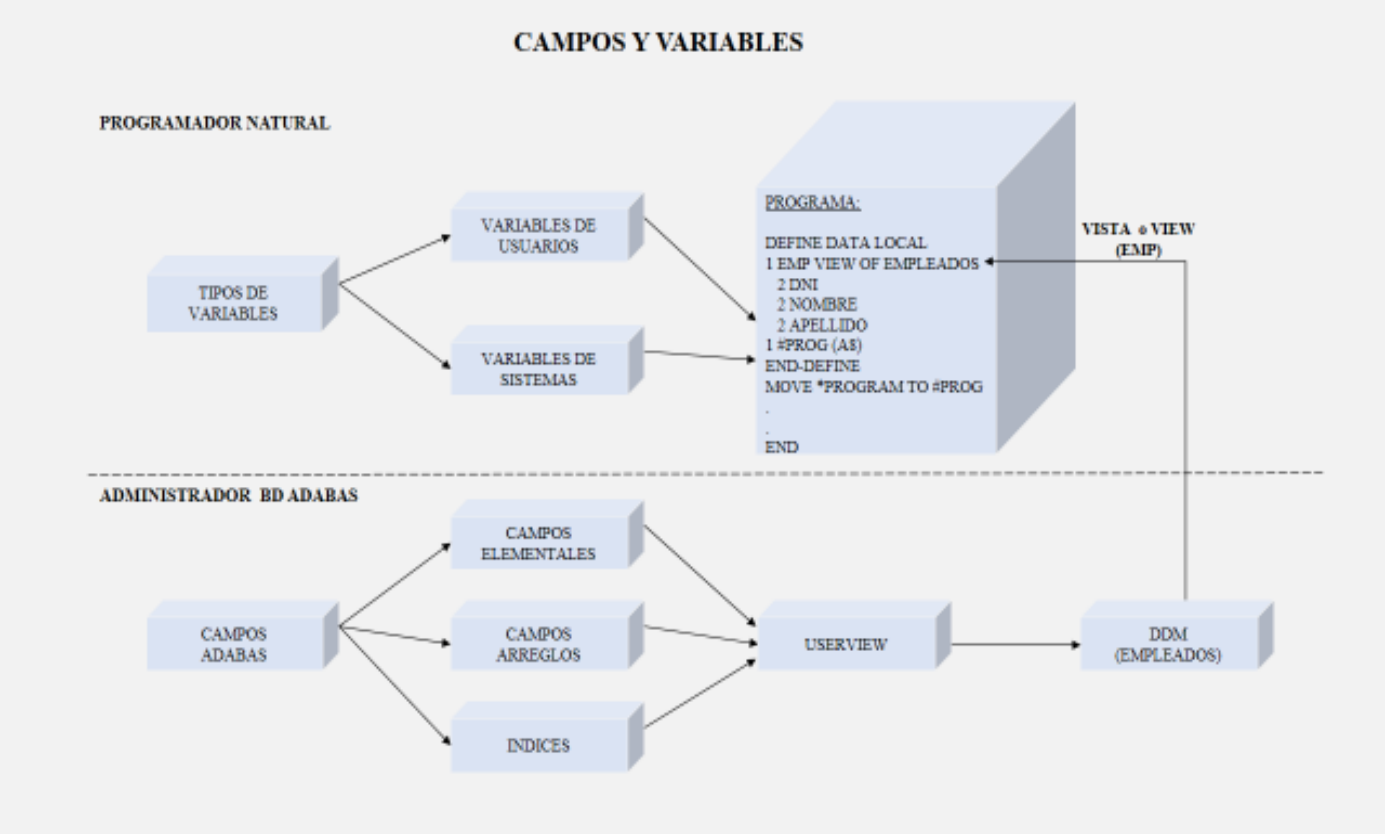
Las relaciones entre entidades son únicamente aquellas que pueden ser representadas usando directorios (índices y listas invertidas). Por ello se considera una BD No-Relacional, aunque simule una BD relacional.

La información de la base de datos se almacena físicamente en tres grandes áreas denominados DATA, ASSO y WORK.

DEFINICIÓN DE DATOS

Este apartado explica de forma sintetizada los diferentes tipos de datos en Natural ADABAS, donde están definidos y como pueden ser implementados en un programa Natural.





Autor: Yorman Guadarrama Martínez

TIPOS DE DATOS:

Tipo	Código Formato
• Alfanumérico	A
• Numérico	N
• Numérico Empaquetado	P
• Binario	B
• Enteros	I
• Punto Flotante	F
• Fecha	D
• Hora	T
• Lógico	L
• Atributo de Control	C

TIPOS DE VARIABLE:

Variable de usuarios: Son las variables locales definidas en un programa y su nombre siempre inician con una letra. Por lo general se define un estándar para iniciarlas con un símbolo (#, V, etc) y diferenciarlas de otros campos del programa.

Variables de Sistemas: Son varíales predefinidas en el sistema Natural Adabas que no pueden ser modificadas y su nombre siempre inician con un "*". Contienen valores como la fecha, hora, programa en ejecución, librería, modo de ejecución (batch o on-line), código de usuario en ejecución, etc.

TIPOS DE CAMPO ADABAS:

Campos Elementales:

➢ Alfanumérico	A
➢ Numérico	N
➢ Numérico Empaquetado	P
➢ Binario	B
➢ Enteros	I
➢ Punto Flotante	F
➢ Fecha	D
➢ Hora	T
➢ Lógico	L

Campos Arreglos:

- **Múltiples (M):** Campos elementales con ocurrencias donde un mismo registro puede contener múltiples valores para dicho campo. Puede tener hasta 65.534 ocurrencias y estad no son significativas para ADABAS ***(aplica supresión estándar)***.

- **Grupos (G):** Agrupación de varios campos elementales.
- **Grupos Periódicos (P):** Agrupación de varios campos elementales con ocurrencias donde un mismo registro puede contener múltiples grupos de valores. Puede tener hasta 65.534 ocurrencias y estas si son significativas para ADABAS ***(no aplica supresión estándar)***. Pueden agrupar tantos campos elementales como se desee.

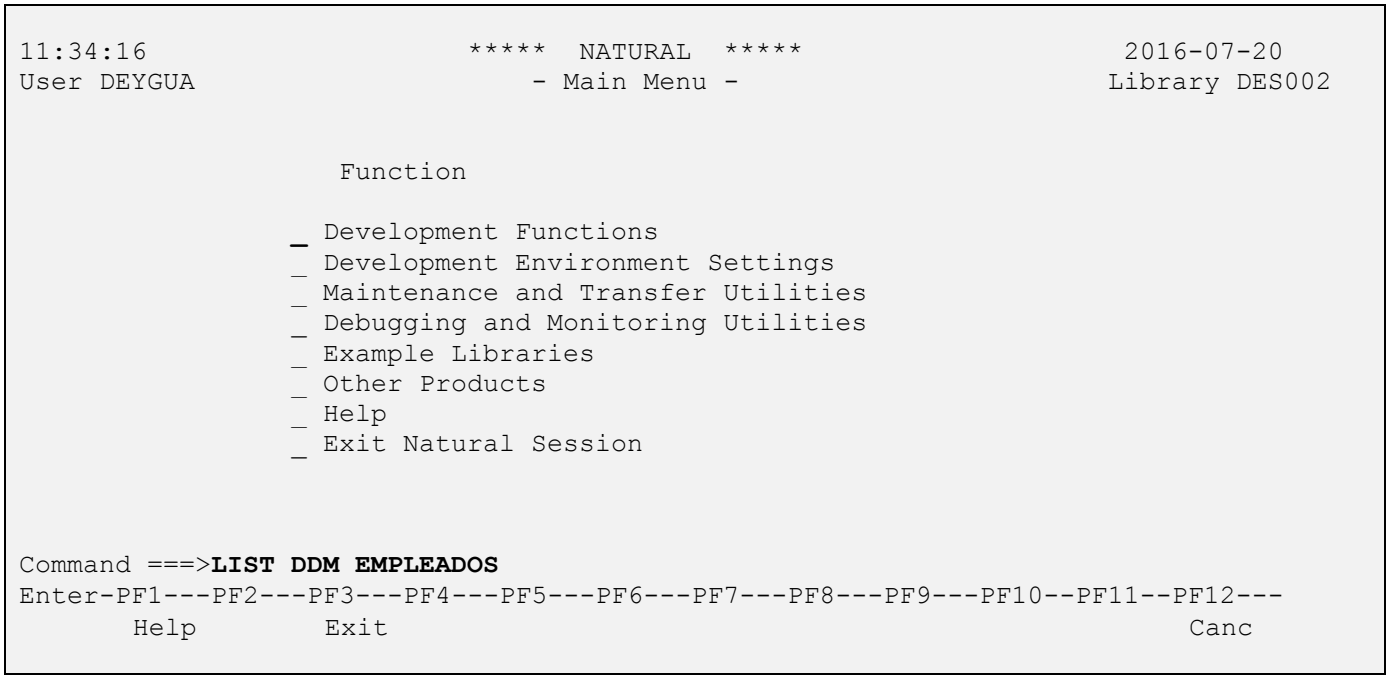
Índices:

- **Descriptor:** Código "**D**" y son campos elementales con un índice asociado el cual puede ser únicos o no.
- **Superdescriptor:** Código "**S**" y son superíndices que están formados por más de un campo Adabas. Estos campos pueden o no ser descriptores.
- **Subdescriptor:** Código "**U**" y son subíndices que están formados por algunos caracteres de uno o varios campos Adabas. Estos campos pueden o no ser descriptores.
- **Descriptor Fonético:** Código "**P**" y son índices fonéticos que optimiza la búsqueda de campos Adabas por su sonido. Por ejemplo el nombre de una persona.

DDM Natural (Data Defination Moduel):

Es una definición lógica de un fichero Adabas y es generada desde PREDICT (Diccionario de datos) como un objeto externo. Es utilizada para hacer referencia de los campos Adabas desde un programa Natural.

- **LIST DDM:** Comando de sistemas utilizado para listar de todas las DDM disponibles para la librería y posteriormente consultar una DDM.
- **LIST DDM *nombre-ddm:*** Se consulta una DDM específica



```
 DB:    0 File:   450 - EMPLEADOS                          Default Sequence

TYL  DB  Name                             F       Leng S D Remark
---  --  -------------------------------- - ---------- - - --------------------
  1  AA  DNI                              A         10 N D
G 1  AB  NOMBRE-COMPLETO
  2  AC  NOMBRE                           A         15 N
  2  AD  PRIMER-APELLIDO                  A         15 N D
  2  AE  SEGUNDO-APELLIDO                 A         15 N
  1  AF  TELEFONO                         N          4 N
  1  AG  ESTADO-CIVIL                     A          1 N   C/S/D/V
M 1  AH  IDIOAS                           A         10     (5)
  1  AI  SEXO                             A          1 N   H-HOMBRE/M-MUJER
  1  AJ  CUMPLE-AÑO                       D          6 N
  1  AK  COD-CARGO                        N          2 N D
P 1  AM  DIRECCION-COMPLETA                             (3)
  2  AN  PROVINCIA                        A         20
  2  AL  CIUDAD                           A         20
  2  AO  CALLE-NRO-PTA                    A         20
  1  AP  SP-CODCARGO-DNI                  A         12   S  SUPER (AK+AA)
Fieldname, '--' Top, 'S' Short, '.' End:
```

T: Tipo de campo.
L: Nivel del campos.
DB: Nombre corto del campo (nombre en Adabas).
Name: Nombre Largo del campo (nombre en Predict).
F: Formato del campo.
Leng: Longitud del campo.
S: Tipo de supresión del campo (blanco: estándar, N: a nulos, F: fija).
D: Si en campo es descriptor (índice) y el tipo de descriptor.
Remark: Comentarios sobre el campo.

DECLARACIÓN DE CAMPOS Y VARIABLES LOCALES:

Con la instrucción **(*DEFINE LADA LOCAL.....END-DEFINE*)** se declaran los campos y variables que serán utilizados en un programa Natural de forma temporal durante su ejecución.

Los campos son referenciados desde una vista Natural, la cual se implementa dentro del DEFINE DATA LOCAL con la instrucción **(*nombre-vista VIEW OF nombre-ddm*)**.

Las variables se declaran dentro del DEFINE DATA LOCAL y aplican solo a variables de usuarios, donde se incluyen las constantes. Las variables de sistemas no requieren ser declaradas, simplemente se usan dentro del cuerpo del programa.

"Cnombre-campo*"** retorna el número de ocurrencias de un campo o variable con ocurrencias.

Para **comentar líneas en un programa Natural** se utilizan los símbolos **"*"** y **"/*"**.

Autor: Yorman Guadarrama Martínez

```
DEFINE DATA LOCAL
1 VIEWEMP VIEW OF EMPLEADOS
  2 NOMBRE                          /* CAMPO ELEMENTAL
  2 DNI                             /* CAMPO ELEMENTAL
  2 IDIOMAS             (5)         /* CAMPO MULTIPLE 5 Occ
  2 NOMBRE-COMPLETO                 /* CAMPO GRUPO
    3 NOMBRE
    3 PRIMER-APELLIDO
    3 SEGUNDO-APELLIDO
  2 DIRECCION-COMPLETA  (2)         /* CAMPO GRUPO PERIODICO
    3 PROVINCIA
    3 CIUDAD
    3 CALLE-NRO-PTA
*
* VARIABLE DE USUARIO
1 #VARI-A    (A20)
1 #VARI-B    (N3.2)
1 #CONSTANTE (N4)     INIT<'2016'>   /* CONSTANTE
1 #ARRAY     (I4/1:3)                /* ARREGLO UNA DIMENSION
1 #ARRAY1    (I4/1:3,1:3)            /* ARREGLO DOS DIMENSIONES
1 #ARRAY3    (I4/1:3,1:3,1:3)        /* ARREGLO TRES DIMENSIONES

END-DEFINE
*
MOVE *PROGRAM TO #NOMBRE-PROGRAMA    /* VARIABLE DE SISTEMAS
*
WRITE 'CANTIDAD DE OCURRENCIAS: ' C*IDIOMAS  /* DEVUELVE LAS Occ DEL CAMPO
```

REDEFINICIÓN DE CAMPOS Y VARIABLES:

La instrucción **REDEFINE** se utiliza para redefinir campos de vistas Natural y variables. Esta instrucción puede ser implementada en todas las áreas de declaración de datos:

- ÁREA LOCAL DE DATOS (LDA).
- ÁREA GLOBAL DE DATOS (GDA).
- ÁREA PARÁMETROS DE DATOS (PDA).

```
DEFINE DATA LOCAL
1 VIEWEMP VIEW OF EMPLEADOS
  2 NOMBRE
  2 DNI
  2 FECHA-INGRESO
  2 REDEFINE FECHA-INGRESO
    3 #ANO   (N4)
    3 #MES   (N2)
    3 #DIA   (N2)
1 #VARI-A    (A20)
1 REDEFINE #VARI-A
  2 #VARI-A-1 (A10)
  2 #VARI-A-2 (A10)
1 #VARI-B    (N3.2)
1 #CONSTANTE (N4)   INIT<'2016'>
END-DEFINE
*
MOVE *PROGRAM TO #NOMBRE-PROGRAMA
```


Autor: Yorman Guadarrama Martínez

OPERACIONES CON VARIABLES

ASIGNACIÓN DE VALORES:

Existen dos formas principales para asignar valores a una variable o campo, usando la instrucción **"MOVE"** o el símbolo **":="**.

```
MOVE 5       TO #VAR1                    ó     #VAR1:= 5
MOVE 5       TO #VAR1 #VAR2
MOVE 'A'     TO #VAR3                    ó     #VAR3:= 'A'
MOVE #VAR1   TO #VAR2                    ó     #VAR2:= #VAR1
MOVE ALL '*' TO #VAR3                    ó     #VAR3:= '**********'

MOVE #ARRAY(#INDICE) TO #VAR1            ó     #VAR1:= #ARRAY(#INDICE)
MOVE #VAR1           TO #ARRAY(#INDICE)  ó     #ARRAY(#INDICE):= #VAR1
MOVE #ARRAY-A(*)     TO #ARRAY-B(*)
```

OPERACIONES ARITMÉTICAS:

SUMA:

```
ADD 3 TO #VAR1                      /* #VAR1 = 3
ADD +5 -1 -2 GIVING #VAR1           /* #VAR1 = 2
COMPUTE #RESUL = #VAR1 + #VAR2      /* #RESULT = #VAR1 + #VAR2
```

RESTA:

```
SUBTRAC 4 FROM #VAR1                /* #VAR1 = #VAR1 – 4
SUBTRAC 4 FROM 10 GIVING #VAR1      /* #VAR1 = 10 -4
COMPUTE #RESUL = #VAR1 - #VAR2      /* #RESULT = #VAR1 - #VAR2
```

MULTIPLICACIÓN:

```
MULTIPLY #VAR1 BY 3                 /* #VAR1 = #VAR1 * 3
MULTIPLY #VAR1 BY 4 GIVING #VAR2    /* #VAR2 = #VAR1 * 4
COMPUTE  #RESUL = #VAR1 * #VAR2     /* #RESULT = #VAR1 * #VAR2
```

DIVISIÓN:

```
DIVIDE 5 INTO #VAR1                 /* #VAR1 = #VAR1 / 5
DIVIDE 5 INTO #VAR1 GIVING #VAR2    /* #VAR2 = #VAR1 / 5
DIVIDE 5 INTO 31 REMAINDER #VAR1    /* #VAR1 = RESTO (31/5)
COMPUTE  #RESUL = #VAR1 / #VAR2     /* #RESULT = #VAR1 / #VAR2
```


Autor: Yorman Guadarrama Martínez

BUCLES

FOR:
Esta instrucción inicia un proceso en bucle el cual es controlado con un índice.

Ejemplo.

```
DEFINE DATA LOCAL
1 #INDEX (I1)
END-DEFINE
*
FOR #INDEX 1 TO 5           /* REALIZA 5 BUCLES CON INDICE DE UNO EN UNO
  WRITE 'INDICE =' #INDEX
END-FOR
*
FOR #INDEX 1 TO 5 STEP 2    /* REALIZA 3 BUCLES CON INDICE DE DOS EN DOS
  WRITE 'INDICE' #INDEX
END-FOR
*
END
```

REPEAT:
Esta instrucción inicia un proceso en bucle con tres modalidades.

Ejemplo.

```
DEFINE DATA LOCAL
1 #CONTADOR (A8)
END-DEFINE
*
REPEAT                   /* BUCLE INFINITO
  ADD 1 TO #CONTADOR
  IF #CONTADOR = 5
     ESCAPE BOTTOM       /* SALE DEL BUCLE
  END-IF
*
  WRITE 'NRO. BUCLE: ' #CONTADOR
END-REPEAT
*
#CONTADOR:= 0
REPEAT WHILE #CONTADOR <= 5       /* BUCLE CONTROLADO CON WHILE
  ADD 1 TO #CONTADOR
  WRITE 'NRO. BUCLE: ' #CONTADOR
END-REPEAT
*
#CONTADOR:= 0
REPEAT                            /* BUCLE CONTROLADO CON UNTIL
  ADD 1 TO #CONTADOR
  WRITE 'NRO. BUCLE: ' #CONTADOR
  UNTIL #CONTADOR = 5
END-REPEAT
*
END
```


Autor: Yorman Guadarrama Martínez

CONDICIONALES

IF:

Esta instrucción se utiliza para condicionar la ejecución de un bloque de instrucciones Natural.

Ejemplo.

```
IF #CONFIRMAR = 'S' THEN
   WRITE 'HA CONFIRMADO QUE REQUIERE AYUDA...'
   ELSE
     IF #CONFIRMAR = 'N' THEN
        WRITE 'HA RECHAZADO LA AYUDA...'
     END-IF
     ELSE IGNORE   /* EN CASO DE SER DISTINTO DE "S" Y "N" IGNORA EL VALOR
END-IF
```

DECIDE FOR:

DECIDE FOR FIRST CONDITION: Solo la primera condición verdadera del ***WHEN*** será procesada.

DECIDE FOR EVERY CONDITION: Todas las condiciones verdaderas del ***WHEN*** serán procesadas.

Ejemplo1.

```
DECIDE FOR FIRST CONDITION
  WHEN #VAR3 = 'A' AND #OP = 1
    WRITE 'LA FUNCION A HA SIDO SELECCIONADA...'
  WHEN #VAR3 = 'B' AND #OP = 2
    WRITE 'LA FUNCION B HA SIDO SELECCIONADA...'
  WHEN #VAR3 = 'C' AND #OP = 3
    WRITE 'LA FUNCION C HA SIDO SELECCIONADA...'
  WHEN #VAR3 = 'D' THRU 'G'
    WRITE 'UNA DE ESTAS FUNCIONES D, E, F o G HA SIDO SELECCIONADA...'
  WHEN NONE
    WRITE 'NO SE HA SELECCIONADO NINGUNA FUNCION...'
END-DECIDE
```

Ejemplo2.

```
DECIDE FOR EVERY CONDITION
  WHEN #VAR1 >= 0
    WRITE '#VAR1 ES UN VALOR POSITIVO...'
  WHEN #VAR1 <= 0
    WRITE '#VAR1 ES UN VALOR NEGATIVO...'
  WHEN ALL
    WRITE '#VAR1 ES CERO...'
  WHEN ANY
    WRITE ALGUNA DE LAS CONDICIONES ANTERIORES ES VERDADERA...'
  WHEN NONE
    IGNORA
END-DECIDE
```

NOTA IMPORTANTE: En cualquier caso las preguntas ***WHEN ANY*** y ***WHEN ALL*** son procesadas pero no son obligatorias. El ***WHEN NONE*** siempre es obligatorio.

DECIDE ON:

DECIDE ON FIRST CONDITION: Solo el primer valor evaluado verdadero del ***WHEN*** será procesado.

DECIDE ON EVERY CONDITION: Todos los valores evaluados verdaderos del ***WHEN*** serán procesados.

Ejemplo1.

```
DECIDE ON FIRST VALUE OF #VAR3
  VALUE 'A'
    WRITE 'LA TECLA A HA SIDO SELECCIONADA...'
  VALUE 'B'
    WRITE 'LA TECLA B HA SIDO SELECCIONADA...'
  ANY VALUE
    WRITE 'ALGUNA TECLA HA SIDO SELECCIONADA...'
  NONE VALUE
    WRITE 'NINGUNA TECLA HA SIDO SELECCIONADA...'
END-DECIDE
```

Ejemplo2.

```
DECIDE ON EVERY VALUE OF #VAR1
  VALUE 1:4
    WRITE 'EL CONTENIDO DE #VAR1 ES 1-4'
  VALUE 2:5
    WRITE ' EL CONTENIDO DE #VAR1 ES 2-5'
  ANY VALUE
    WRITE ' EL CONTENIDO DE #VAR1 ES 1-5'
  ALL VALUE
    WRITE ' EL CONTENIDO DE #VAR1 ES 2-4'
  NONE VALUE
    WRITE ' EL CONTENIDO DE #VAR1 NO ESTA ENTRE 1-5'
END-DECIDE
```

NOTA IMPORTANTE: En ambos ejemplos el ***ANY VALUE*** y el ***ALL VALUE*** son procesados pero no son preguntas obligatorias. El ***NONE VALUE*** siempre es obligatorio.

ÁREA LOCAL DE DATOS (LDA): Es un área donde se han definido los campos que serán usados en un Objeto (Programa, Subprograma, Subrutina Externa o Helprutina). Estos campos pueden ser datos de vistas Natural, variables de usuario o constantes.

Una LDA puede ser definida de forma interna o externa a estos objetos.

LDA Interna: Solo está disponible en el objeto donde ha sido definida.

Programa:

```
DEFINE DATA LOCAL
1 VIEWEMP VIEW OF EMPLEADOS
  2 APELLIDO
  2 NOMBRE
  2 DNI
1 #VARI-A (A20)
1 #VARI-B (N3.2)
1 #VARI-C (I4)
END-DEFINE
...
```

LDA Externa: Esta disponible para múltiples objetos, al estar definida un objeto externo. Pero sus datos solo son actualizados por el objeto que inicia la ejecución y en consecuencia instancia la LDA.

Un objeto puede hacer referencia a más de una LDA.

La LDA y los objetos deben estar creados en la misma librería.

Permite un fácil mantenimiento y en general se usa en aplicaciones estructurada.

Programa Objeto:

```
DEFINE DATA LOCAL
      USING LDA39
END-DEFINE
  ...
```

Objeto LDA: LDA39

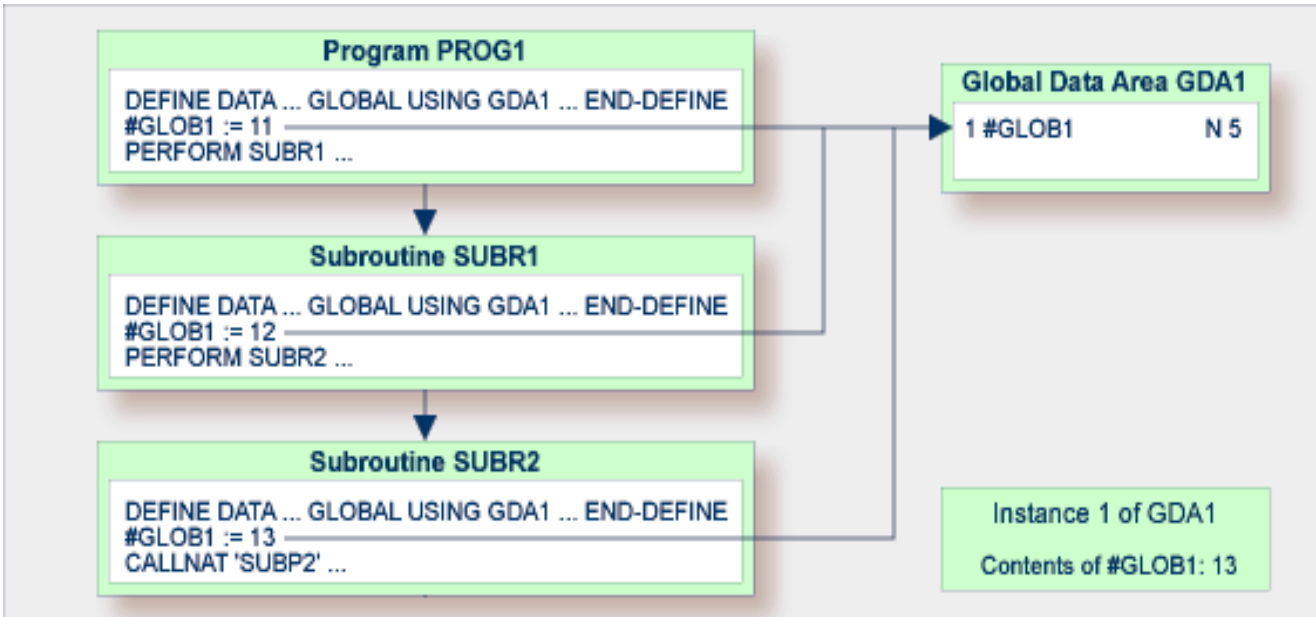
```
I T L  Name                            F Length     Miscellaneous
All -- ------------------------------- - ---------- ------------------------->
  V  1 VIEWEMP                                      EMPLEADOS
     2 DNI                             A          8
     2 NOMBRE                          A         20
     2 APELLIDO                        A         20
     1 #VARI-A                         A         20
     1 #VARI-B                         N        3.2
     1 #VARI-C                         I          4
```


Autor: Yorman Guadarrama Martínez

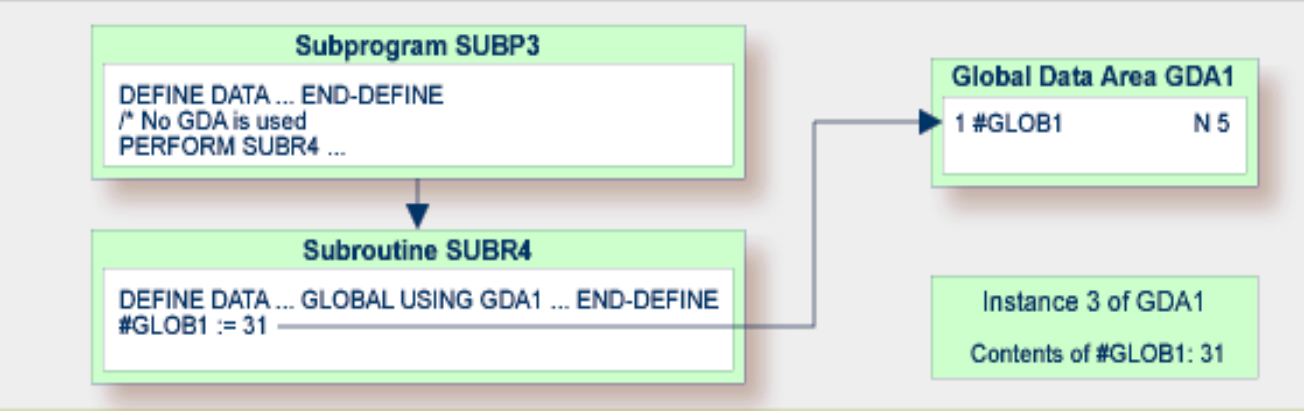
ÁREA GLOBAL DE DATOS (GDA): Es un área de datos similar a la LDA con las siguientes diferencias básicas:

- Una GDA siempre se define externa a los objetos Natural.
- Un objeto solo puede referenciar a una sola GDA.
- Permite la actualización de sus datos desde los todos los objetos que la referencian cuando es instanciada.

Ejemplo 1.



Ejemplo 2.

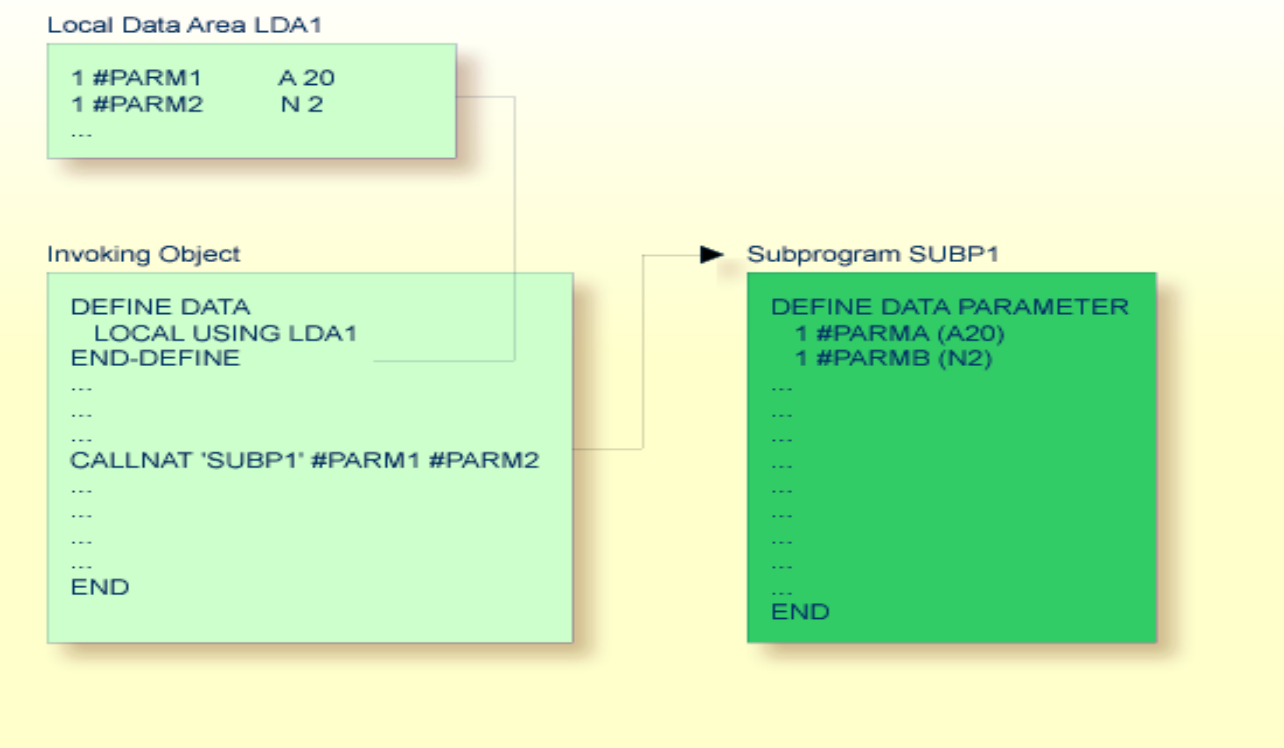


Autor: Yorman Guadarrama Martínez

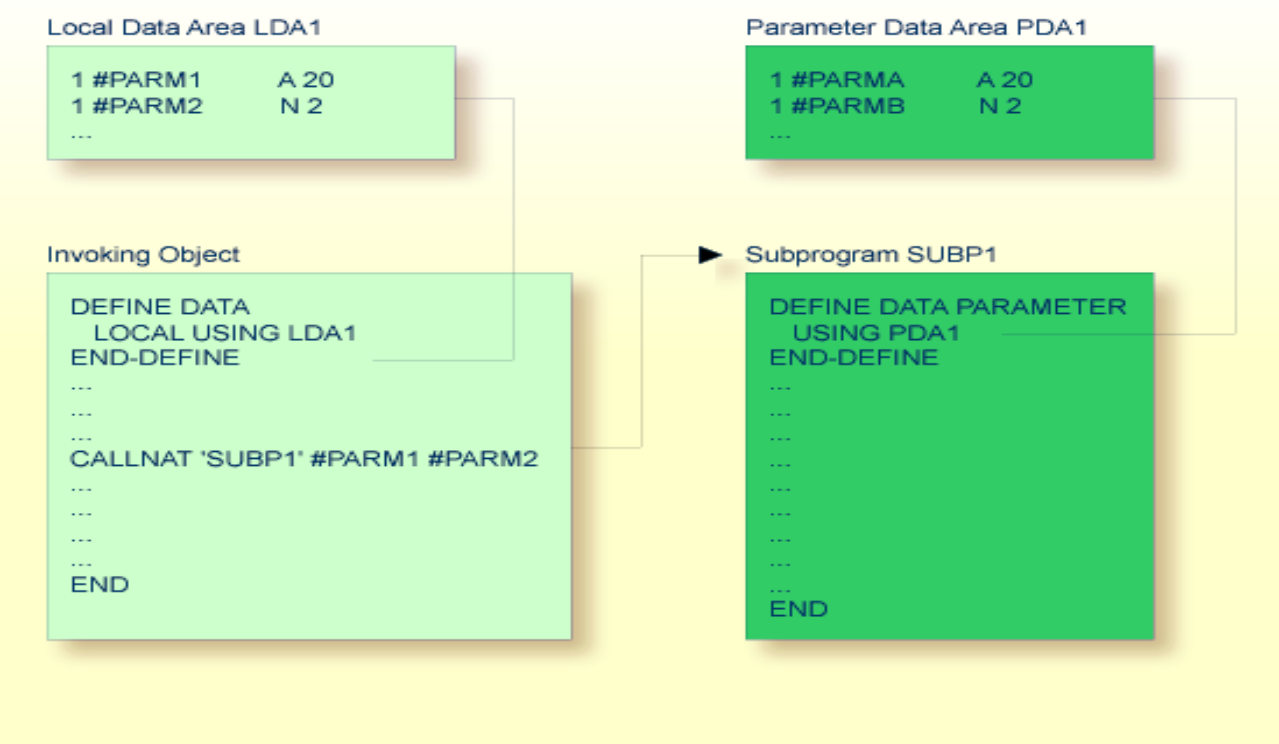
ÁREA DE PARÁMETROS DE DATOS (PDA): Es un área de datos definida para recibir parámetros en un objeto que ha sido llamado desde otro (Programa, Subprograma, Subrutina Externa o Helprutina).

Una PDA puede ser definida de forma interna o externa a estos objetos.

PDA Interna: Solo está disponible en el objeto donde ha sido definida.

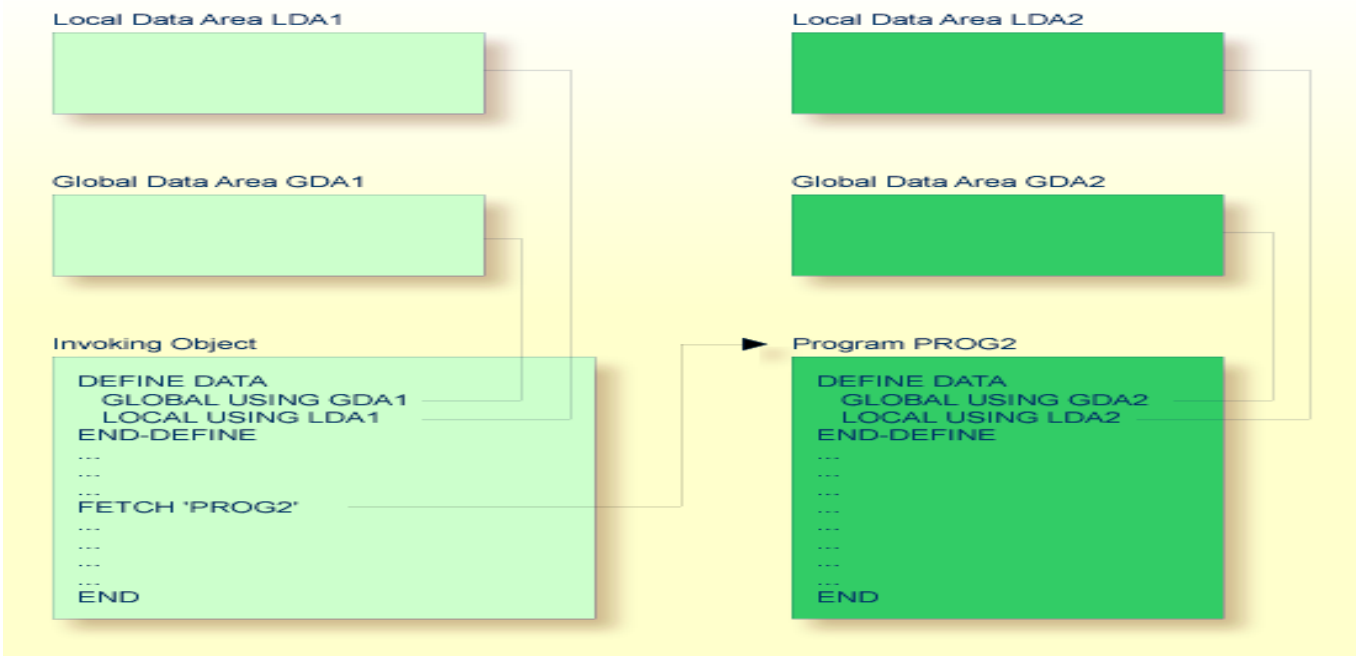


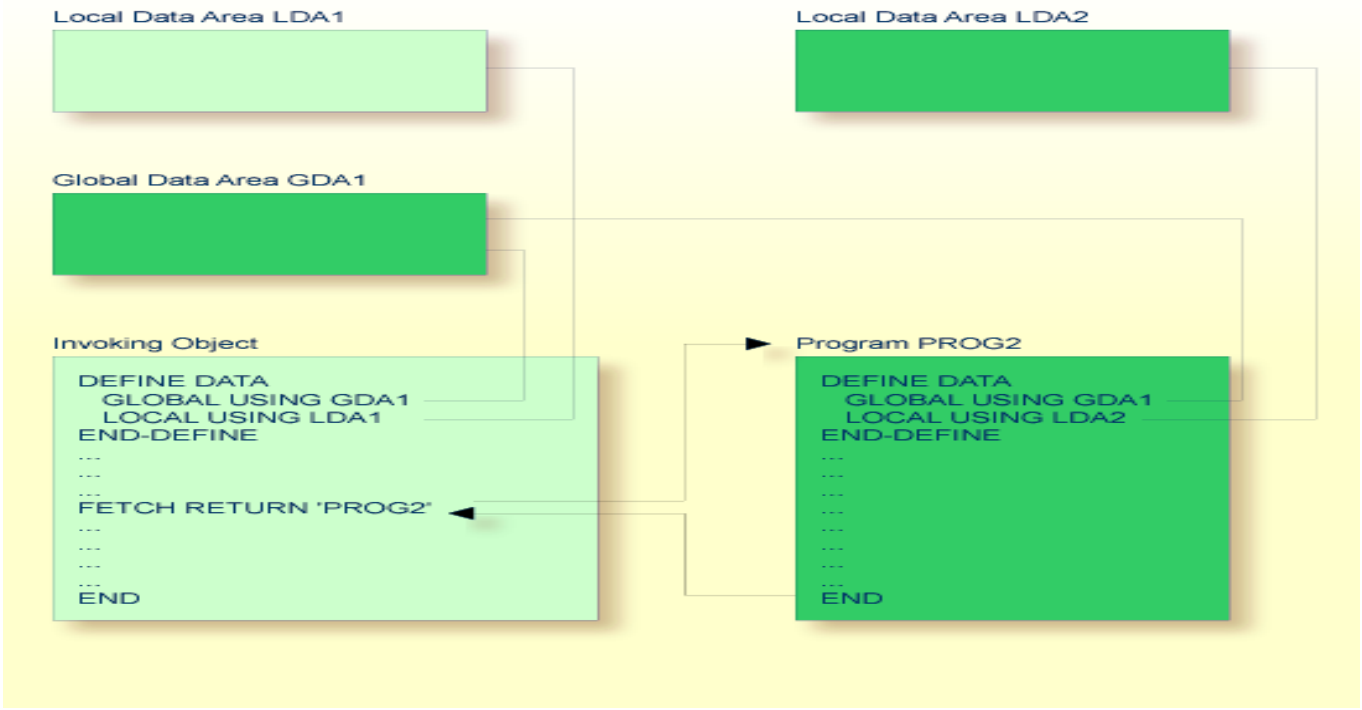
PDA Externa: Esta disponible para múltiples objetos.



Autor: Yorman Guadarrama Martínez

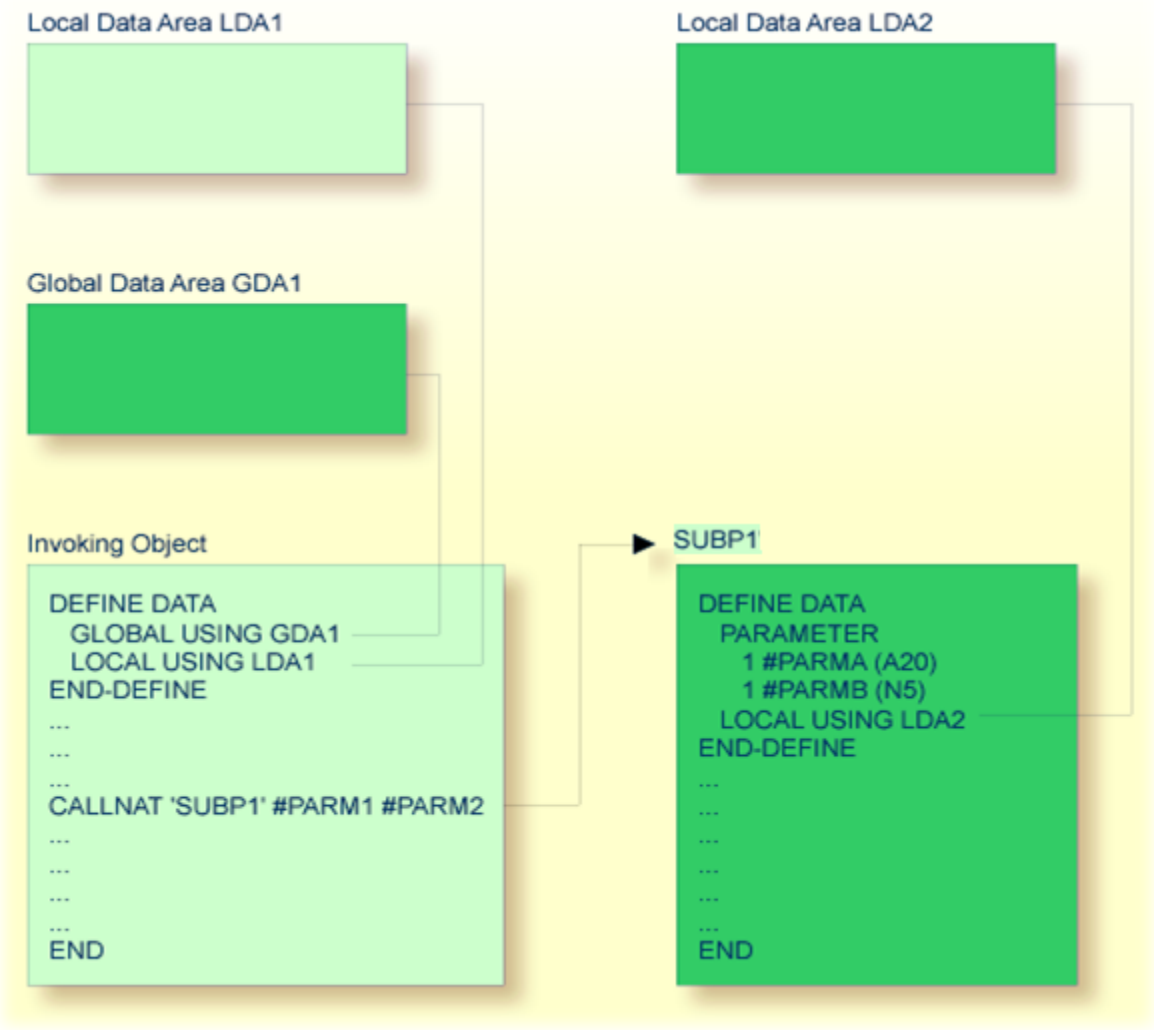
PROGRAMA: Es un **Programa Natural** y puede ser considerado como el objeto principal de una aplicación desarrollada bajo Natural Adabas y puede ser ejecutado en modo on-line o batch.
Un Programa puede ser ejecutado directamente sin que otro objeto lo llame, con los comandos de sistemas **RUN** o **EXECUTE.** RUN es utilizado para ejecutar el Programa fuente y EXECUTE para ejecutar Programa Objeto (compilado).
Un Programa también puede ser ejecutado desde otro Programa utilizando las instrucciones **FETCH** o **FETCH RETURN.** Con FETCH el proceso termina cuando se ha ejecutado el 100% del programa llamado y con FETCH RETURN una vez culmine esta ejecución, el control regresa al programa llamante para continuar ejecutando el resto de las líneas de código.





Autor: Yorman Guadarrama Martínez

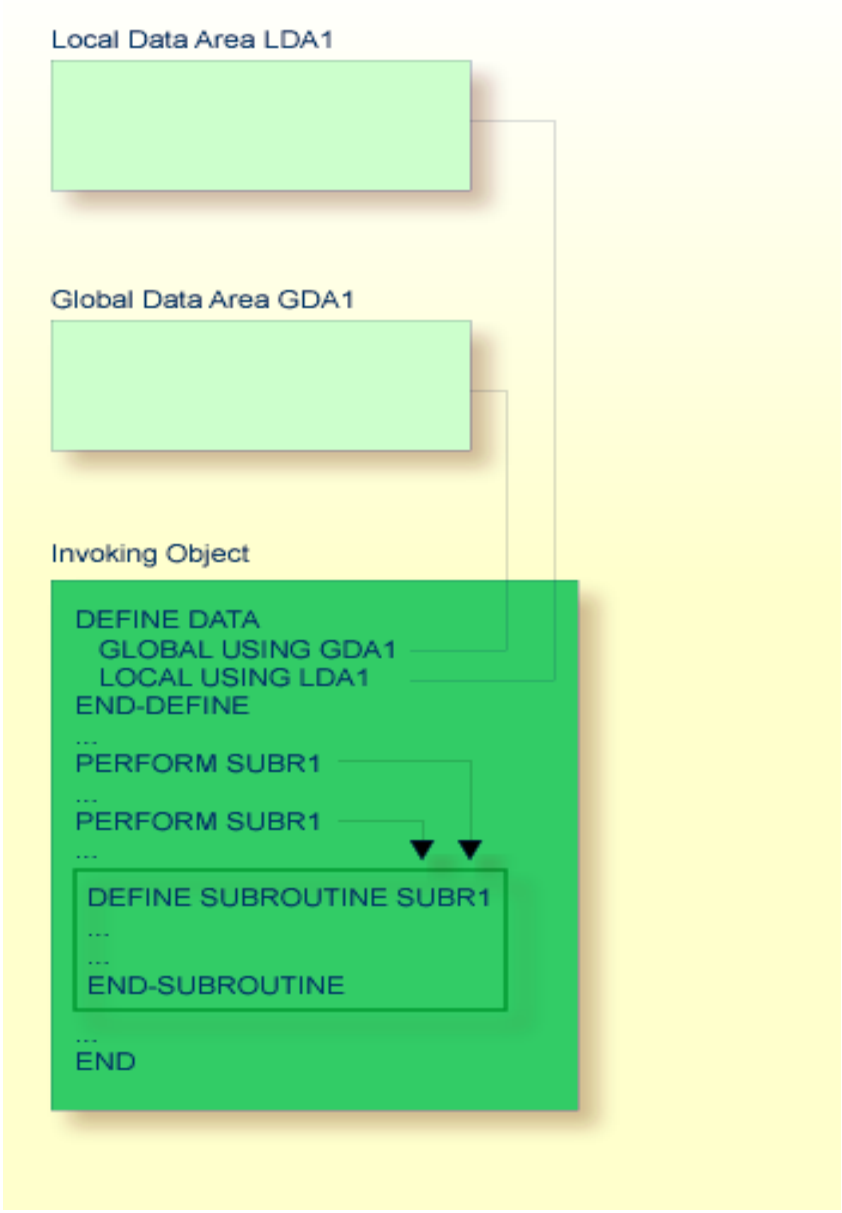
SUBPROGRAMA: Es un Programa compilado que solo puede ser ejecutado desde otro objeto y nunca por sí mismo. Por lo tanto es un objeto individual. Para ser ejecutado se utiliza la instrucción **CALLNAT.**



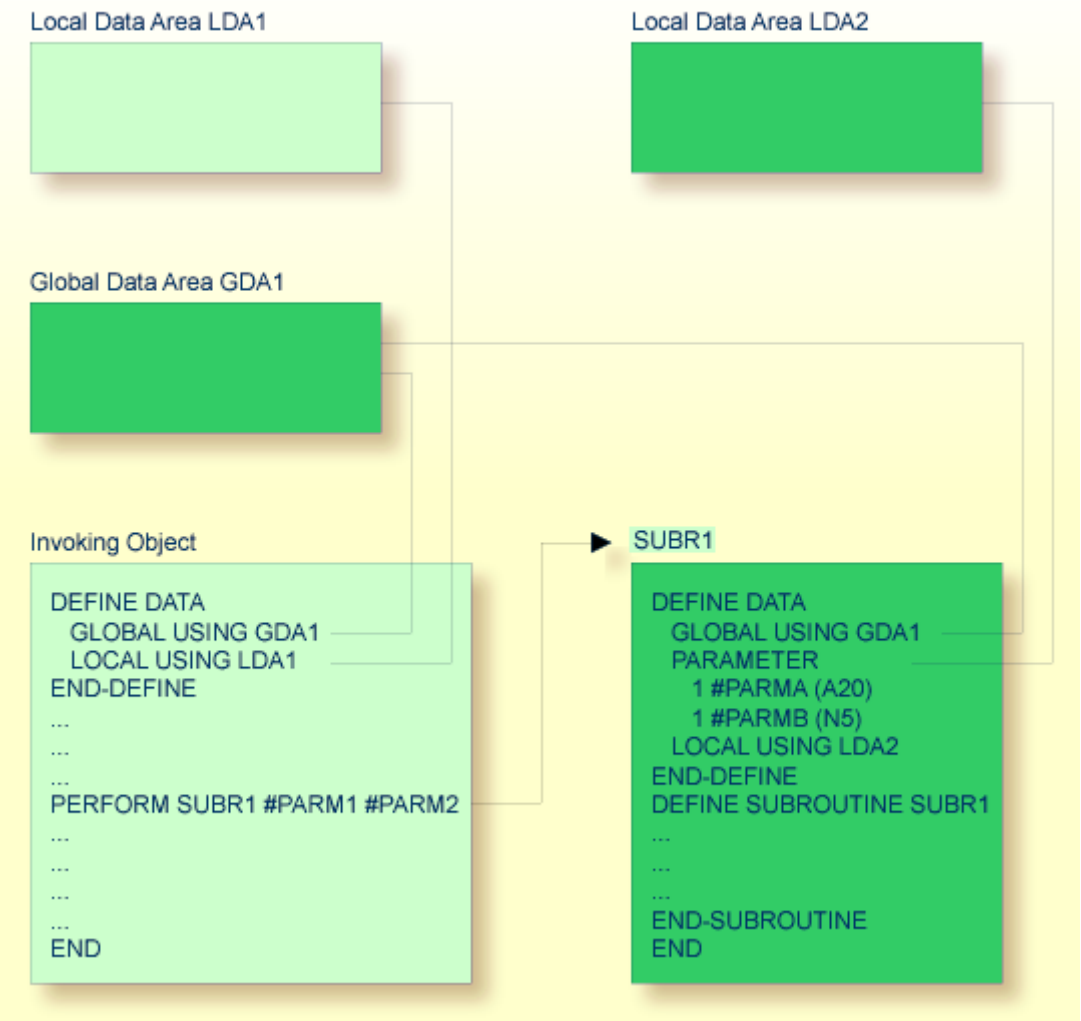
Autor: Yorman Guadarrama Martínez

SUBRUTINA: Normalmente es un ***bloque de código Natural*** que se encuentra adentro de un Programa para ser ejecutado múltiples veces por este **(Subrutina Interna).**
Si la Subrutina requiere ser ejecutada una solo vez por uno Programa el bloque estará definido fuera de este como un objeto individual **(Subrutina Externa).**
Para ejecutar una Subrutina se usa la instrucción **PERFORME** y el bloque de código se definen dentro de la instrucción **DEFINE SUBROUTINE *nombre-subrutina.....END-SUBROUTINE.***
Una Subrutina no puede ser ejecutada por sí misma, solo por otro objeto.

Subrutina Interna:



Subrutina Externa:



HELPRUTINA: Es similar a la Subrutina pero con características específicas. Es utilizada para implementar sistemas de ayuda complejos e interactivos a través de cuadros emergentes en una aplicación en Natural Adabas. Es un objeto que no puede ser ejecutada por sí misma.

Normalmente una Helprutina es implementada con la tecla PF1 o con el símbolo "?", a través del diseño de un objeto tipo **MAPA** y las instrucciones utilizadas por este.

MAPA: Este tipo de objeto es utilizado para diseñar pantallas dinámicas. Un MAPA es llamado o ejecutado desde un programa NATURAL usando la instrucción **UNPUT USING MAP *nombre-mapa***. No puede ser ejecutado por sí mismo.

Autor: Yorman Guadarrama Martínez

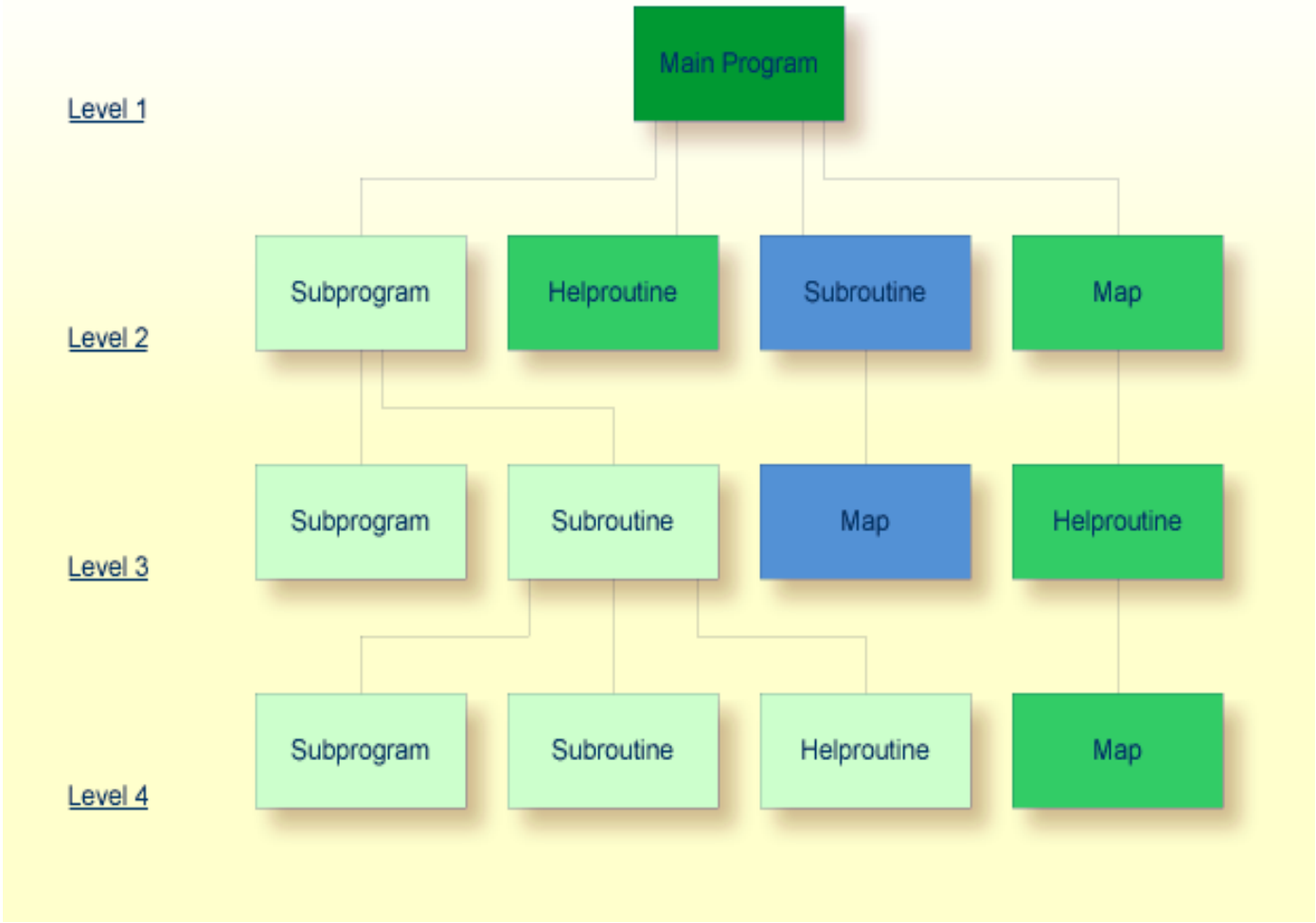
COPYCODE: Es una porción de código fuente que se encuentra fuera de un Objeto y es incluida en este para ser ejecutada con la instrucción **INCLUDE.** Un COPYCODE no puede ser ejecutado por sí mismo y no es un objeto individual ya que no puede ser compilado, solo salvado. Es incluido en el objeto cuando este es compila.

Se utiliza para simplificar la codificación y puede ser usado por distintos objetos, pero no es la norma.

TEXTO: No es realmente un objeto ya que no se compila solo se salva. Normalmente se usa para documentar objetos Natural con más detalle. No puede ser ejecutado, solo mostrado en un editor. El TEXTO se usa cuando el PREDICT no está disponible para la documentación de los programas.

Niveles al ejecutar múltiples objetos: Los objetos tipo Programas, Subprogramas, Subrutinas y Helprutina, pueden ser llamados desde otros Programas, Subprogramas, Subrutinas o Helprutina. Esto genera un nivel en para cada objeto ejecutado.

La variables de sistema ***LEVEL** es utilizada para saber en qué nivel un objeto ejecutado dentro de una aplicación en Natural Adabas.



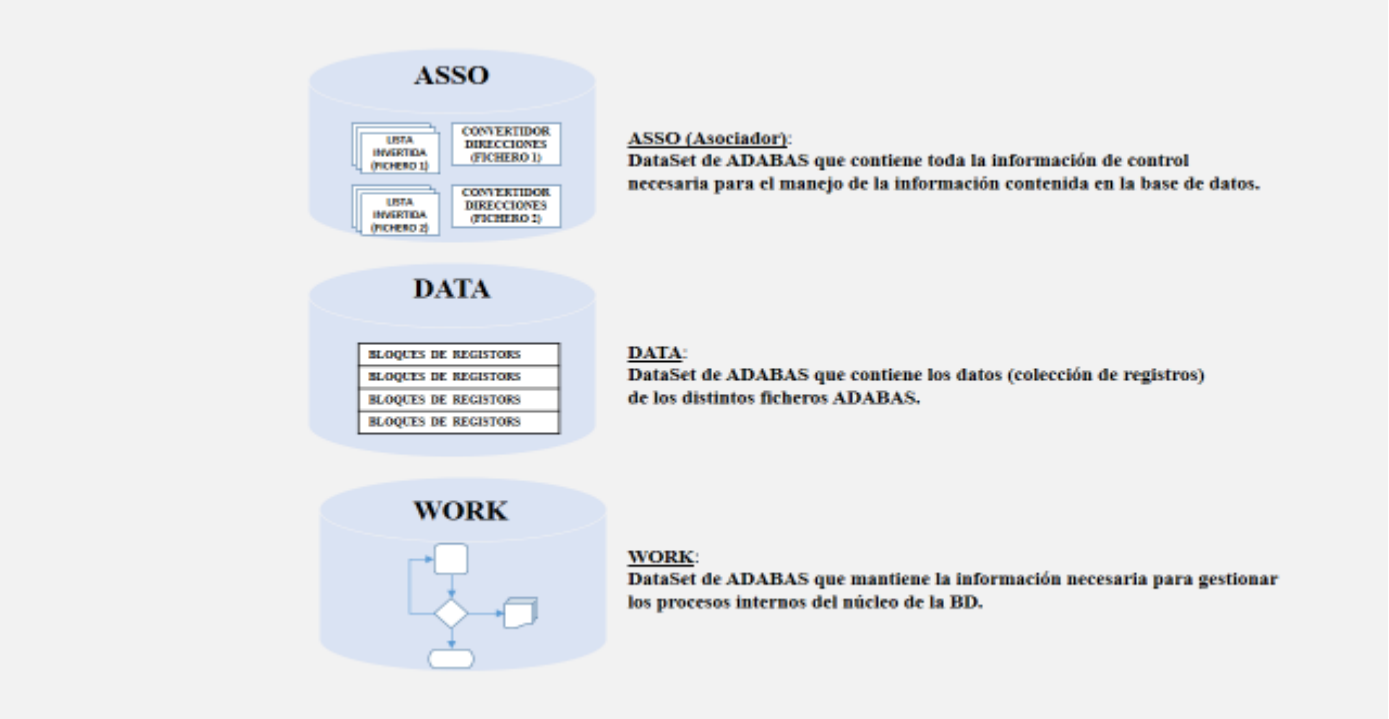
Autor: Yorman Guadarrama Martínez

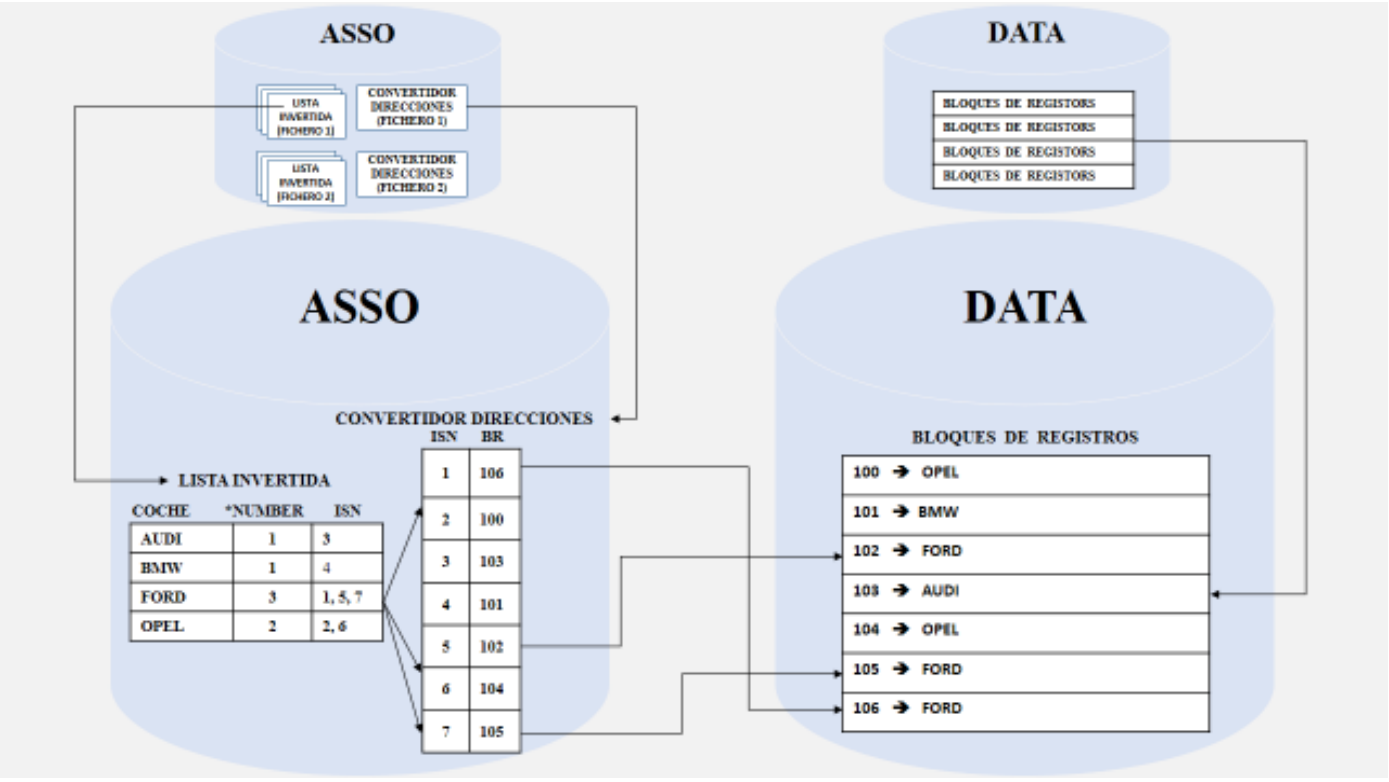
ACCESOS A LA BASE DE DATOS ADABAS

Una base de datos ADABAS está formada principales por 3 áreas (***ASSO, DATA y WORK***) .

El área ASSO contiene dos tipos de tablas:

- **Lista Invertida:** Existe una por cada campos descriptor de un fichero Adabas y contiene el valor del campo descriptor, la cantidad de registros que contienen ese valor y el ISN de cada registro.
- **Convertidor de Direcciones:** Existe uno por cada fichero Adabas y es donde se asocia cada ISN con el número del bloque del registro (***RABN***) en el **área DATA.**





Autor: Yorman Guadarrama Martínez

READ:

La instrucción READ se utiliza para leer los registros de una base de datos. Los registros se pueden leer en secuencia física, en la secuencia del ISN o en la secuencia del valor de un campo descriptor (clave). La instrucción READ inicia un procesamiento en bucle.

Nota: *El* READ *por su comportamiento dentro del gestor de base de datos, es recomendado para la recuperación de altos volúmenes de registros.*

```
{ READ   }  [ { ALL          } ]   [MULTI-FETCH-clause] [RECORDS] [IN] [FILE] view-name
{ BROWSE }  [ { (operand1)   } ]

          [PASSWORD=operand2]

          [CIPHER=operand3]

          [WITH REPOSITION]


          [sequence/range-specification]


          [STARTING WITH ISN=operand4]

          [WHERE logical-condition]

          statement ...

END-READ  (structured mode only)

[LOOP]    (reporting mode only)
```


Autor: Yorman Guadarrama Martínez

```
*
* READ (Ejemplos 1, 2, 3 y 4)
*
*  VIEW y VARIABLES:
*
DEFINE DATA LOCAL
*
1 EMPLEADOS-VIEW VIEW OF EMPLEADOS
  2 DNI              /* DESCRIP
  2 APELLIDO         /* DESCRIP
  2 NOMBRE
  2 SALARIO
  2 CIUDAD           /* DESCRIP
  2 FUNCIONES (10)
  2 COD-CARGO        /* DESCRIP
  2 SP-CODCARGO-DNI  /* SUPER (COD-CARGO + DNI)
*
1 VEHICULOS-VIEW VIEW OF VEHICULOS
  2 DNI            /* DESCRIP
  2 MATRICULA
  2 MARCA
*
1 #VARI-A (A20) INIT<'M'>      /* ALFANUMERICA
1 #VARI-B (N3)                 /* NUMERICA
1 #VARI-C (N9.2)               /* NUMERICA CON DECIMAL
1 #VARI-D (A5/1:5)             /* ARRAY (1 DIMENSION y 5 OCURRENCIAS)
1 #VARI-E (A5/1:5,1:5)         /* ARRAY (2 DIMENSION y 5 OCURRENCIAS)
1 #VARI-F (L)                  /* Bolean
1 #SP-CODCARGO-DNI (A12)
1 REDEFINE #SP-CODCARGO-DNI
  2 #COD-CARGO     (A10)
  2 #DNI           (N2)
*
END-DEFINE
*
LIMIT 3     /* LIMITA 3 LINEAS POR PAGINA
*
WRITE 'READ EN SECUENCIA FISICA:'
READ EMPLEADOS-VIEW IN PHYSICAL SEQUENCE
  DISPLAY NOTITLE DNI APELLIDO *ISN *COUNTER
END-READ
*
WRITE / 'READ EN ORDEN DE ISN:'
READ EMPLEADOS-VIEW BY ISN STARTING FROM 2 ENDING AT 3
  DISPLAY NOTITLE DNI APELLIDO *ISN *COUNTER
END-READ
*
WRITE / 'READ EN ORDEN AL DESCRIPTOR APELLIDOS:'
READ EMPLEADOS-VIEW BY APELLIDO
  DISPLAY NOTITLE DNI APELLIDO *ISN *COUNTER
END-READ
*
WRITE / 'READ EN ORDEN AL DESCRIPTOR APELLIDOS INICIANDO DESDE LA (M):'
READ EMPLEADOS-VIEW BY APELLIDO STARTING FROM #VARI-A
  DISPLAY NOTITLE DNI APELLIDO *ISN *COUNTER
END-READ
*
LIMIT 10
WRITE / 'READ EN ORDEN DEL SUPERDESCRIPTOR:'
MOVE ALL '0' TO #COD-CARGO
MOVE ALL ' ' TO #DNI
READ EMPLEADOS-VIEW BY #SP-CODCARGO-DNI
  DISPLAY NOTITLE COD-CARGO DNI APELLIDO *ISN *COUNTER
END-READ
END
```

SALIDA EJEMPLOS 1,2,3 y 4 (READ)

```
   DNI          APELLIDOS          ISN         CNT
---------- -------------------- ----------- -----------

READ EN SECUENCIA FISICA:
50005800   ADAM                           1           1
50005600   MORENO                         2           2
50005500   BLOND                          3           3

READ EN ORDEN DE ISN:
50005600   MORENO                         2           2
50005500   BLOND                          3           3

READ EN ORDEN AL DESCRIPTOR APELLIDOS:
60008339   ABELLAN                      478           1
30000231   ALVAREZ                      878           2
50005800   ADAM                           1           3

READ EN ORDEN AL DESCRIPTOR APELLIDOS INICIANDO DESDE LA (M):
30008125   MARTINEZ                     923           1
20028700   MENDEZ                       765           2
50005600   MORENO                         2           3
```

SALIDA EJEMPLOS 5 (READ)

```
COD CARGO    DNI          APELLIDOS          ISN         CNT
--------- ---------- -------------------- ----------- -----------

READ EN ORDEN DEL SUPERDESCRIPTOR:
        1 30000231   ALVAREZ                      878           1
        1 50005500   BLOND                          3           2
        1 60008339   ABELLAN                      478           3
        2 30008125   MARTINEZ                     923           4
        2 50005600   MORENO                         2           5
        3 11400319   ROBLEDO                       93           6
        3 20028700   MENDEZ                       765           7
        3 50005800   ADAM                           1           8
        4 30034045   PEREZ                        125           9
```


Autor: Yorman Guadarrama Martínez

```
*
* READ...REPOSITION (Ejemplos 6)
*
DEFINE DATA LOCAL
1 EMPLEADOS-VIEW VIEW OF EMPLEADOS
  2 DNI
  2 APELLIDO
  2 NOMBRE
  2 SALARIO
*
1 #STARTVAL (A20) INIT <'A'>
1 #ATTR     (C)
END-DEFINE
...
LIMIT 3
SET KEY PF3
...
READ EMPLEADOS-VIEW WITH REPOSITION BY APELLIDO = #STARTVAL
*
  DISPLAY NOTITLE DNI APELLIDO *ISN *COUNTER
*
  INPUT (IP=OFF AD=O) 'Apellido: ' APELLIDO //
    'Nuevo valor de inicio para la reposición:' #STARTVAL (AD=MT CV=#ATTR)/
    'PF3 para detener el READ...'
  IF *PF-KEY = 'PF3'
    THEN STOP
  END-IF
  IF #ATTR MODIFIED
    THEN ESCAPE TOP REPOSITION
  END-IF
*
END-READ
*
END
```

SALIDA EJEMPLOS 6 (READ)

```
   DNI          APELLIDOS          ISN         CNT
--------- -------------------- ----------- -----------

60008339  ABELLAN                     478           1
50005800  ADAM                          1           3
30000231  ALVAREZ                     878           2

Apellido: ABELLAN
Nuevo valor de inicio para la reposición:
```


Autor: Yorman Guadarrama Martínez

FIND:

La instrucción FIND se utiliza para seleccionar un conjunto de registros de la base de datos, basado en un criterio de búsqueda con campos definidos como descriptores (claves).

Esta instrucción inicia un proceso en bucle y luego ejecuta la recuperación de cada registro seleccionado.

Cada campo de cada registro puede ser referenciado dentro de dicho bucle.

Nota: *El* FIND *por su comportamiento dentro del gesto de base de datos, es recomendado para la recuperación de uno o un número reducido de registros.*

```
                ALL
              [ ( (operand1) ) ]
FIND          [ { FIRST      } ]  [MULTI-FETCH-clause] [RECORDS] [IN] [FILE] view-name
              [ { NUMBER     } ]
              [ ( UNIQUE     ) ]

          [PASSWORD=operand2]

          [CIPHER=operand3]

          [WITH] [[LIMIT] (operand4)] basic-search-criterion

          [COUPLED-clause] ... 4/42

          [STARTING WITH ISN=operand5]]

          [SORTED-BY-clause]

          [RETAIN-clause]

          [WHERE-clause]

          [IF-NO-RECORDS-FOUND-clause]

          statement ...

END-FIND                        (structured mode only)

[LOOP]                          (reporting mode only)
```


Autor: Yorman Guadarrama Martínez

```
*
* FIND (Ejemplos 1)
*
DEFINE DATA LOCAL
*
1 EMPLEADOS-VIEW VIEW OF EMPLEADOS
  2 DNI
  2 APELLIDO
  2 NOMBRE
  2 CIUDAD
*
WRITE 'Criterio por DNI:'
FIND EMPLEADOS-VIEW WITH DNI = '20008400'
     DISPLAY NOTITLE DNI APELLIDO *ISN *NUMBER *COUNTER
END-FIND
*
WRITE / 'Criterio por APELLIDO:'
FIND EMPLEADOS-VIEW WITH APELLIDO = 'PEREZ'
     DISPLAY NOTITLE DNI APELLIDO *ISN *NUMBER *COUNTER
END-FIND
*
WRITE / 'Criterio por CIUDAD:'
FIND EMPLEADOS-VIEW WITH CIUDAD = 'MADRID'
     DISPLAY NOTITLE DNI APELLIDO *ISN *NUMBER *COUNTER
END-FIND
*
END
```

SALIDA EJEMPLOS 1 (FIND)

```
   DNI          APELLIDOS          ISN         NMBR       CNT
--------- -------------------- ----------- ----------- -----------

Criterio por DNI:
20008400  LOPEZ                        329           1           1

Criterio por APELLIDO:
20003400  PEREZ                        643           2           2
30034045  PEREZ                        125           2           3

Criterio por CIUDAD:
20008400  LOPEZ                        329           3           1
50005600  MORENO                       518           3           2
20003400  PEREZ                        643           3           3
```


Autor: Yorman Guadarrama Martínez

```
*
* READ...FIND...IF NO RECORD FOUND  (Ejemplos 2)
*
DEFINE DATA LOCAL
1 EMPLEADOS-VIEW VIEW OF EMPLEADOS
  2 DNI
  2 APELLIDO
  2 NOMBRE
  2 SALARIO
  2 CIUDAD
*
1 VEHICULOS-VIEW VIEW OF VEHICULOS
  2 DNI
  2 MATRICULA
  2 MARCA
*
END-DEFINE
*
LIMIT 10
*
RD. READ EMPLEADOS-VIEW BY APELLIDO STARTING FROM 'MORENO'
*
  FD. FIND VEHICULOS-VIEW WITH DNI = DNI (RD.)
    IF NO RECORDS FOUND
       WRITE '** NO EXISTE **'
    END-NOREC
    DISPLAY NOTITLE (ES=OFF IS=ON ZP=ON AL=15)
            DNI (RD.)
            APELLIDO (RD.)
            NOMBRE (RD.)
            CIUDAD (RD.)
            MARCA (FD.) (IS=OFF)
  END-FIND
*
END-READ
*
END
```

SALIDA EJEMPLOS 2 (FIND)

DNI	APELLIDO	NOMBRE	CIUDAD	MARCA
50005600	MORENO	EDWARD	MADRID	GENERAL MOTORS
20002000	OROPEZA	MARTA	BARCELONA	GENERAL MOTORS
20003400	PEREZ	LAURA	MADRID	GENERAL MOTORS
30034045	PEREZ	KEVIN	MURCIA	TOYOTA
30034233	RAMIREZ	GREGORY	LONDRES	FORD
11400319	ROBLEDO	MANUEL	BARCELONA	** NO EXISTE **

```
*
* FIND...IF *NUMBER  (Ejemplos 3)
*
DEFINE DATA LOCAL
*
1 EMPLEADOS-VIEW VIEW OF EMPLEADOS
  2 DNI
  2 APELLIDO
  2 NOMBRE
  2 SALARIO
  2 CIUDAD
*
FIND EMPLEADOS-VIEW WITH CIUDAD = 'TOLEDO' THRU 'VALENCIA'
*
     IF *NUMBER = 0
         WRITE 'No Existen Empleados...'
     ELSE
         DISPLAY NOTITLE DNI APELLIDO CIUDAD *ISN *NUMBER
     END-IF
*
     ......
*
END-FIND
*
END
```

SALIDA EJEMPLOS 3 (FIND)

IF *NUMBER = 0 ➔ SI NO EXISTEN EMPLEADOS EN CIUDADES DESDE TOLEDO HASTA VALENCIALA LA SALIDA DEL PROGRAMA SERÁ: **(No Existen Empleados...)**

ELSE ➔ DE LO CONTRARIO LA SALISA SERÁ:

```
      DNI          APELLIDO         CIUDAD        NMBR
--------------- --------------- --------------- -----------
20028700        MENDEZ          VALENCIA                  1
```


Autor: Yorman Guadarrama Martínez

```
*
* READ y FIND Combinados  (Ejemplos 4)
*

DEFINE DATA LOCAL
1 EMPLEADOS-VIEW VIEW OF EMPLEADOS
  2 DNI
  2 APELLIDO
  2 NOMBRE
  2 SALARIO
  2 CIUDAD
*
1 VEHICULOS-VIEW VIEW OF VEHICULOS
  2 DNI
  2 MATRICULA
  2 MARCA
*
END-DEFINE
*
LIMIT 10
*
RD. READ EMPLEADOS-VIEW BY APELLIDO STARTING FROM 'JONES'
*
  FD. FIND VEHICULOS-VIEW WITH DNI = DNI (RD.)
    IF NO RECORDS FOUND
      ENTER
    END-NOREC
    DISPLAY NOTITLE (ES=OFF IS=ON ZP=ON AL=15)
            DNI (RD.)
            APELLIDO (RD.)
            NOMBRE (RD.)
            CIUDAD (RD.)
            MARCA (FD.) (IS=OFF)
  END-FIND
*
END-READ
*
END
```

SALIDA EJEMPLOS 4 (FIND)

DNI	APELLIDO	NOMBRE	CIUDAD	MARCA
20007500	JONES	VIRGINIA	NUEVA YORK	CHRYSLER
20008400	LOPEZ	MARTA	MADRID	CHRYSLER
				CHRYSLER
30008125	MARTINEZ	ROBERTO	LAS PALMAS GC	GENERAL MOTORS
20028700	MENDEZ	LILLY	VALENCIA	FORD
				AUDI
50005600	MORENO	EDWARD	MADRID	GENERAL MOTORS
20002000	OROPEZA	MARTA	BARCELONA	GENERAL MOTORS
20003400	PEREZ	LAURA	MADRID	GENERAL MOTORS
30034045	PEREZ	KEVIN	MURCIA	TOYOTA
30034233	RAMIREZ	GREGORY	LONDRES	FORD
11400319	ROBLEDO	MANUEL	BARCELONA	

Autor: Yorman Guadarrama Martínez

HISTOGRAM:

La instrucción HISTOGRAM es utiliza para leer los valores de un campos definidos como descriptor, superdescriptor o subdescriptor de la base de datos.

Los valores son leídos directamente desde las listas invertidas de ADABAS.

Esta instrucción inicia un proceso en bucle, pero no tiene acceso a ningún campo de la base de datos que no haya sido definido en la propia instrucción.

```
HISTOGRAM  [ ALL          ]   [MULTI-FETCH-clause ] [multi-fetch-factor]
           [ (operand1)   ]   [IN] [FILE] view-name

     [PASSWORD=operand2]

                          ASCENDING
        [                 {          }                          ]
          [IN]            { DESCENDING }  [SEQUENCE]
                          { VARIABLE operand3 }
        [                 { DYNAMIC  operand3 }                 ]

  [VALUE] [FOR] [FIELD] operand4

  [STARTING/ENDING-clause]

  [WHERE logical-condition]

   statement ...

END-HISTOGRAM (structured mode only)

[LOOP] (reporting mode only)
```


Autor: Yorman Guadarrama Martínez

```
*
* HISTOGRAM (Ejemplos 1)
*
*  VIEW y VARIABLES:
*
DEFINE DATA LOCAL
*
1 EMPLEADOS-VIEW VIEW OF EMPLEADOS
  2 DNI
  2 APELLIDO
  2 NOMBRE
  2 SALARIO
  2 CIUDAD
*
END-DEFINE
*
HISTOGRAM EMPLEADOS-VIEW CIUDAD STARTING FROM 'M'
  DISPLAY NOTITLE CIUDAD 'NUMERO DE/EMPEADOS' *NUMBER *COUNTER
END-READ
*
END
```

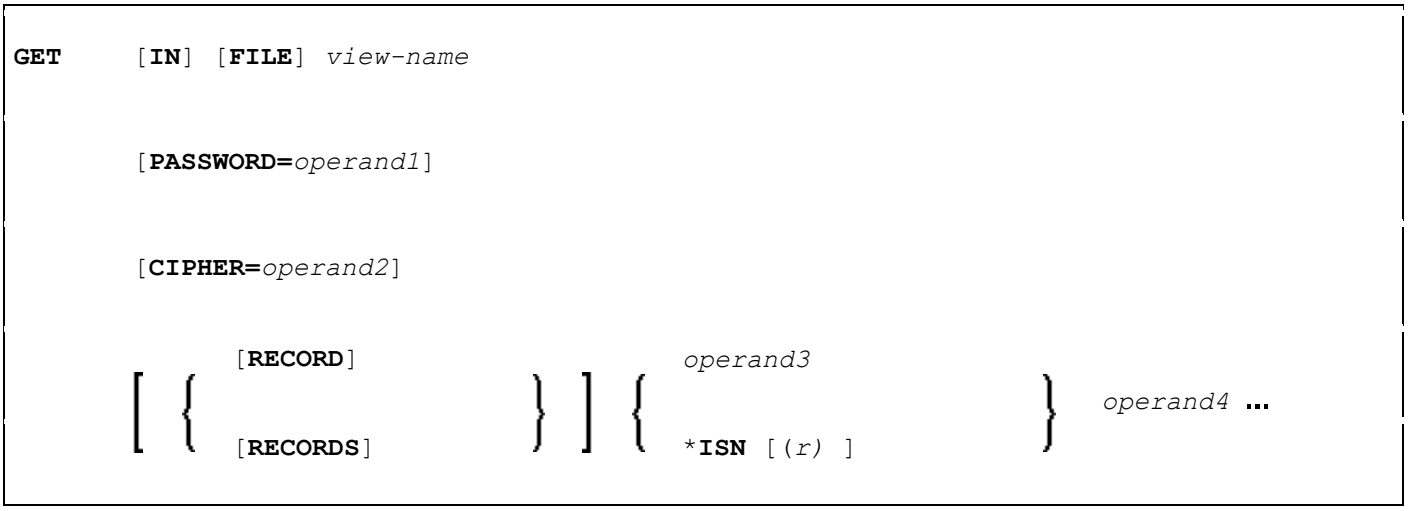
SALIDA EJEMPLOS 1 (HISTOGRAM)

CIUDAD	NUMERO DE EMPLEADOS	CNT
MADRID	3	1
MURCIA	1	2
NUEVA YORK	1	3
VALENCIA	1	4

GET:

La instrucción GET es usada para **leer y bloquear** un único registro con un **ISN** dado. Esta instrucción no inicia un procesamiento en bucle.

NOTA: *El* GET *por su comportamiento dentro del gestor de base de datos, es recomendado para la actualización on-line de un registro en una base de datos ADABAS.*



Autor: Yorman Guadarrama Martínez

```
*
* GET  (Ejemplos 1)
*
DEFINE DATA LOCAL
1 EMPLEADOS-VIEW VIEW OF EMPLEADOS
  2 DNI
  2 APELLIDO
  2 NOMBRE
  2 CIUDAD
*
1 SALARIO-VIEW VIEW OF EMPLEADOS
  2 SALARIO
*
1 #ISN  (B4)
*
END-DEFINE
*
FORMAT PS=16   /* Ancho de pagina
LIMIT 10
*
RD. READ EMPLEADOS-VIEW BY APELLIDO STARTING FROM 'JONES'
  MOVE *ISN TO #ISN
  DISPLAY DNI APELLIDO NOMBRE CUIDAD
*
  GET SALARIO-VIEW #ISN
    WRITE   'SALARIO EMPLEADO: ' SALARIO
    /
*
END-READ
END
```

SALIDA EJEMPLOS 1 (GET)

```
      DNI          APELLIDO         NOMBRE          CIUDAD
--------------- --------------- --------------- ---------------
20007500        JONES           VIRGINIA        NUEVA YORK
SALARIO EMPLEADO: 1245

20008400        LOPEZ           MARTA           MADRID
SALARIO EMPLEADO: 1385

30008125        MARTINEZ        ROBERTO         LAS PALMAS GC
SALARIO EMPLEADO: 1198

20028700        MENDEZ          LILLY           VALENCIA
SALARIO EMPLEADO: 1204

50005600        MORENO          EDWARD          MADRID
SALARIO EMPLEADO: 1220

20002000        OROPEZA         MARTA           BARCELONA
SALARIO EMPLEADO: 1317

20003400        PEREZ           LAURA           MADRID
SALARIO EMPLEADO: 1245

30034045        PEREZ           KEVIN           MURCIA
SALARIO EMPLEADO: 1263

30034233        RAMIREZ         GREGORY         LONDRES
SALARIO EMPLEADO: 1230

11400319        ROBLEDO         MANUEL          BARCELONA
SALARIO EMPLEADO: 1176
```

GET SAME:

La instrucción GET SAME se utiliza para volver a leer el registro que se está procesando en el bucle.

Su uso es más frecuente cuando se necesita obtener los valores de campos múltiples o grupos periódicos (campos con ocurrencias), siempre que dichas ocurrencias existan.

```
GET SAME [(r)]
```

GET (Ejemplos 2)

```
READ DIRECCION-EMP-VIEW BY APELLIDO
  WRITE // 12T #APELLIDO
  WRITE /  12T LINEA-DIRECCION (I.1)
*
  FOR I = 2 TO #NRO-LINEAS
     GET SAME                      /* LEER LA SIGUIENTE OCCURRENCIA
     WRITE 12T LINEA-DIRECCION (I.1)
  END-FOR
*
  WRITE / CODIGO-POSTAL CUIDAD
END-READ
```

GET TRANSACTION DATA:

La instrucción GET TRANSACTION DATA se utiliza para leer registros que previamente han sido actualizados con la instrucción END TRANSACTION.

Esta instrucción no inicia un procesamiento en bucle.

```
GET TRANSACTION [DATA]  operand1 ...
```

```
MOVE ' ' TO #DNI
GET TRANSACTION DATA #DNI
*
REPEAT
*
  INPUT  10X 'INDIQUE EL DNI QUE REQUIERE MODIFICAR:' #DNI
*
  FIND EMPLEADOS-VIEW WITH DNI = #DNI
    IF NO RECORDS FOUND
      REINPUT 'NO EXISTEN REGISTROS ENCONTRADOS'  MARK 1 ALARM
    END-NOREC
    INPUT (AD=M) DNI (AD=O)
                / APELLIDO
                / NOMBRE
                / CIUDAD
    UPDATE
    END TRANSACTION #DNI
  END-FIND
*
END-REPEAT
```


Autor: Yorman Guadarrama Martínez

STORE:

La instrucción STORE se utiliza para insertar registros en la base de datos.

```
STORE    [RECORD] [IN] [FILE] view-name

         [PASSWORD=operand1]

         [CIPHER=operand2]

         [    [          ]                ]
         [    [ USING    ]                ]
         [    [          ]  NUMBER operand3]
         [    [ GIVING   ]                ]
```


Autor: Yorman Guadarrama Martínez

```
*
* STORE  (Ejemplos 1)
*
DEFINE DATA LOCAL
1 EMPLEADOS-VIEW VIEW OF EMPLEADOS
  2 DNI
  2 APELLIDO
  2 NOMBRE
  2 SALARIO
  2 CIUDAD
*
1 #DNI       (A10)
1 #APELLIDO  (A20)
1 #NOMBRE    (A15)
1 #SALARIO   (N9.2)
1 #CIUDAD    (A15)
1 #CONFIRMAR (A1) INIT <'N'>
*
END-DEFINE
*
REPEAT
   INPUT  'CONFIRMAR QUE EL EMPLEADO NO EXISTE:' #DNI
*
   FIND NUMBER EMPLEADOS-VIEW WITH DNI = #DNI
   IF *NUMBER > 0
      REIMPUT 'EMPLEADO CON DNIQUE YA EXISTE.'  MARK 1 ALARM
   ELSE
      INPUT
        'DNI             :' #DNI /
        'APELLIDO        :' #APELLIDO /
        'NOMBRE          :' #NOMBRE /
        'SALARIO         :' #SALARIO /
        'CIUDAD          :' #CIUDAD  /
   END-IF
*
   EMPLEADOS-VIEW.DNI      := #DNI
   EMPLEADOS-VIEW.APELLIDO := #APELLIDO
   EMPLEADOS-VIEW.NOMBRE   := #NOMBRE
   EMPLEADOS-VIEW.SALARIO  := #SALARIO
   EMPLEADOS-VIEW.CIUDAD   := #CIUDAD

   STORE EMPLEADOS-VIEW
   END TRANSACTION
*
   INPUT 'CONTINUAR CREANDO EMPLEADOS (S/N):' #CONFIRMAR (AD=M) /
   IF #CONFIRMAR = 'N'
      ESCAPE BOTTOM
   END-IF
*
END-REPEAT
END
```


Autor: Yorman Guadarrama Martínez

UPDATE:

La instrucción UPDATE se utiliza para actualizar uno o varios campos de un registro que ya exista en la base de datos.

El registro previamente debe haber sido seleccionado con una instrucción **FIND, READ o GET.**

Con la ejecución del UPDATE el registro entra en estado "**HOLD**" y cuando esto ocurre no está disponible para otro usuario. Si otro usuario intenta seleccionar el registro entra en estado "**WAIT**" hasta que es liberado con un instrucción END TRANSACTION o BACKOUT TRANSACTION. Si el registro no es liberado y se ha excedido el parámetro **WH** (Wait for Record in Hold Status) será retornado un mensaje de error al usuario que espera por el registro.

NOTA: *En un **proceso on-line** se recomienda el uso de las instrucciones* **GET y END TRANSACTION** *cuando se han actualizado los campos de un registro con la instrucción* **UPDATE,** *siempre que se ha actualizado uno o un número reducido de registros.*

*En caso de **procesos batch** donde se pretende actualizar un número elevado de registros se recomienda solo la instrucción* **END TRANSACTION.**

*De esta forma aseguramos una **base de datos consistente.***

```
UPDATE  [RECORD] [IN] [STATEMENT] [(r)]
```


Autor: Yorman Guadarrama Martínez

```
*
* UPDATE  Procedo On-line (Ejemplos 1)
*
DEFINE DATA LOCAL
1 EMPLEADOS-VIEW VIEW OF EMPLEADOS
  2 DNI
  2 APELLIDO
  2 NOMBRE
  2 SALARIO
  2 CIUDAD
*
1 #DNI       (A10)
1 #APELLIDO  (A20)
1 #NOMBRE    (A15)
1 #SALARIO   (N9.2)
1 #CIUDAD    (A15)
1 #CONFIRMAR (A1) INIT <'N'>
END-DEFINE
*
R1. REPEAT
   INPUT 'INDIQUE DNI DEL EMPLEADO A MODIFICAR: ' #DNI
*
   F1. FIND EMPLEADOS-VIEW WITH DNI = #DNI    /* SELECCIONAR REG.
      IF NO RECORD FOUND
         REINPUT 'DNI DEL EMPLEADO NO EXISTE INTENTE DE NUEVO' MARK 1 ALARM
      END-NOREC
      IF *NUMBER = 1
         MOVE *ISN(F1.) TO #ISN
         /*
         #DNI      := EMPLEADOS-VIEW.DNI
         #APELLIDO := EMPLEADOS-VIEW.APELLIDO
         #NOMBRE   := EMPLEADOS-VIEW.NOMBRE
         #SALARIO  := EMPLEADOS-VIEW.SALARIO
         #CIUDAD   := EMPLEADOS-VIEW.CIUDAD
      ELSE
         REINPUT 'DNI DEL EMPLEADO ESTA DUPLICADO' MARK 1 ALARM
      END-IF
   END-FIND
*
   INPUT (AD=M)
         'DNI             :' #DNI (AD=0) /
         'APELLIDO        :' #APELLIDO /
         'NOMBRE          :' #NOMBRE /
         'SALARIO         :' #SALARIO /
         'CIUDAD          :' #CIUDAD  /
*
   /* REINPUT (VALIDACIONES DE LOS VARIABLE DEL INPUT)...
*
   G1. GET EMPLEADOS-VIEW #ISN                /* LEER Y BLOQUEAR REG.
*
   EMPLEADOS-VIEW.DNI      := #DNI
   EMPLEADOS-VIEW.APELLIDO := #APELLIDO
   EMPLEADOS-VIEW.NOMBRE   := #NOMBRE
   EMPLEADOS-VIEW.SALARIO  := #SALARIO
   EMPLEADOS-VIEW.CIUDAD   := #CIUDAD
*
   UPDATE (G1.)                               /* ACTUALIZAR REG.
   END TRANSACTION                            /* CONSOLIDAR ACTUALIZACION.
*
   INPUT 'CONTINUAR MODIFICANDO EMPLEADOS (S/N):' #CONFIRMAR (AD=M) /
   IF #CONFIRMAR = 'N'
      ESCAPE BOTTOM(R1.)
   END-IF
END-REPEAT
END
```


Autor: Yorman Guadarrama Martínez

```
*
* UPDATE  Procedo Batch(Ejemplos 2)
*
DEFINE DATA LOCAL
1 EMPLEADOS-VIEW VIEW OF EMPLEADOS
  2 APELLIDO
  2 SALARIO
  2 FUNCIONES(10)
*
1 #SALARIO     (N9.2) INIT <1450.50>
1 #CONT-UPDATE (N2)   INIT <0>
END-DEFINE
*
R1. READ EMPLEADOS-VIEW BY APELLIDO           /* LEER TODOS LOS REGISTROS
*
   IF FUNCIONES(1) = 'PROGRAMADOR SENIOR'
      EMPLEADOS-VIEW.SALARIO  := #SALARIO
      UPDATE (R1.)
      ADD 1 TO #CONT-UPDATE
   END-IF
*
   IF #CONT >= 50
      END TRANSACTION                         /* CONSOLIDAR ACTUALIZACION.
      RESET #CONT-UPDATE
   END-IF
*
END-READ
*
END
```


Autor: Yorman Guadarrama Martínez

DELETE:

La instrucción DELETE se utiliza para borrar un registro de la base de datos. Se puede implementar con las instrucciones **FIND y READ.**

Con la instrucción DELETE ocurre una situación similar al UPDATE. Los registros procesados entran en estado **"HOLD"** y no son liberados hasta que se ha ejecuta un END TRANSACTION. Pero en este caso no se requiere de la ejecución del GET.

NOTA: *Para borrar registros en* ***procesos on-line y batch****, se recomienda la misma consideración y técnica usada en el UPDATE.*

```
DELETE [RECORD] [IN] [STATEMENT] [(r)]
```


Autor: Yorman Guadarrama Martínez

```
*
* DELETE  Procedo On-line (Ejemplos 1)
*
DEFINE DATA LOCAL
1 EMPLEADOS-VIEW VIEW OF EMPLEADOS
  2 DNI
  2 APELLIDO
  2 NOMBRE
  2 SALARIO
  2 CIUDAD
*
1 #DNI       (A10)
1 #CONFIRMAR (A1) INIT <'N'>
1 #SN        (A1) INIT <'N'>
END-DEFINE
*
REPEAT
   INPUT 'INDIQUE DNI DEL EMPLEADO A ELIMINAR: ' #DNI
*
   F1. FIND EMPLEADOS-VIEW WITH DNI = #DNI    /* SELECCIONAR REG.
      IF NO RECORD FOUND
         REINPUT 'DNI DEL EMPLEADO NO EXISTE INTENTE DE NUEVO' MARK 1 ALARM
      END-NOREC
      IF *NUMBER = 1
         DISPLAY NOTITLE
            EMPLEADOS-VIEW.DNI
            EMPLEADOS-VIEW.APELLIDO
            EMPLEADOS-VIEW.NOMBRE
            EMPLEADOS-VIEW.SALARIO
            EMPLEADOS-VIEW.CIUDAD
            //
      ELSE
         REINPUT 'DNI DEL EMPLEADO ESTA DUPLICADO' MARK 1 ALARM
      END-IF
*
      INPUT 'DESEA ELIMINAR EL EMPLEADO SELECCIONADO: ' #SN
      IF #SN = 'N'
         ESCAPE BOTTOM
      ELSE
         DELETE (F1.)                  /* BORRAR REG.
         END TRANSACTION               /* CONSOLIDAR ACTUALIZACION.
      END-IF
   END-FIND
*
   INPUT 'CONTINUAR ELIMINANDO EMPLEADOS (S/N):' #CONFIRMAR (AD=M) /
   IF #CONFIRMAR = 'N'
      ESCAPE BOTTOM
   END-IF
*
END-REPEAT
END
```


Autor: Yorman Guadarrama Martínez

```
*
* DELETE  Procedo Batch(Ejemplos 2)
*
DEFINE DATA LOCAL
1 EMPLEADOS-VIEW VIEW OF EMPLEADOS
  2 APELLIDO
  2 SALARIO

*
1 #SALARIO     (N9.2) INIT <2000.00>
1 #CONT-UPDATE (N2)   INIT <0>
END-DEFINE
*
R1. READ EMPLEADOS-VIEW BY APELLIDO           /* LEER TODOS LOS REGISTROS
*
   IF EMPLEADOS-VIEW.SALARIO >= #SALARIO
      DELETE (R1.)
      ADD 1 TO #CONT-UPDATE
   END-IF
*
   IF #CONT >= 50
      END TRANSACTION                         /* CONSOLIDAR ELIMINACION.
      RESET #CONT-UPDATE
   END-IF
*
END-READ
*
END
```


Autor: Yorman Guadarrama Martínez

END TRASACTION y BACKOUT TRANSACTION:

END TRANSACTION: Es utilizado para finalizar y confirmar la ejecución de una transacción logia (ejemplo: STORE, UPDATE, DELETE).

Una transacción lógica es la unidad lógica más pequeña de trabajo, que **debe ser ejecutada para asegurar que la información contenida en la base de datos este lógicamente consistente.**

Si el END TRANSACTION finaliza satisfactoriamente, se está asegurando que todas las actualizaciones lógicas han sido aplicadas físicamente en la base de datos. De lo contrario todas estas actualizaciones serán deshechas automáticamente.

END TRANSACTION también libera todos registros que se encuentren en estado HOLD (Retenidas) como resultado de la ejecución de una actualización lógica de la base de datos.

BACKOUT TRANSACTION: Es utilizado para retirar o dar marcha atrás todas las actualizaciones lógicas que han sido aplicadas físicamente en la base de datos. Liberando todos los registros que han sido retenidos con estado HOLD, para que vuelvan a estar disponibles a otros usuarios. Aseguran así que los datos este lógicamente consistente.

```
END [OF]TRANSACTION   [operand1 ...]
```

```
BACKOUT [TRANSACTION]
```


Autor: Yorman Guadarrama Martínez

WRITE WORK FILE:

La instrucción WRITE WOKR FILE es utilizada para escribir un registro en un WORK FILIE (fichero secuencial) en secuencia física y no inicia procesamiento el bucle.

Esta instrucción solo puede ser ejecutada desde un programa natural en modo batch y bajo TSO, CICS, Com-Plete y otros. Lo más usual es que sea ejecutado desde un JCL bajo TSO.

El programa batch es ejecutado desde el JCL, que es donde se ha definido el formato y longitud del WORK FILE.

En el programa batch se definen los datos que serán insertados en el WORK FILE y que probablemente tengan como origen la lectura de un fichero ADABAS.

WRITE WORK [**FILE**] *work-file-number* [**VARIABLE**] *operand1* **...**

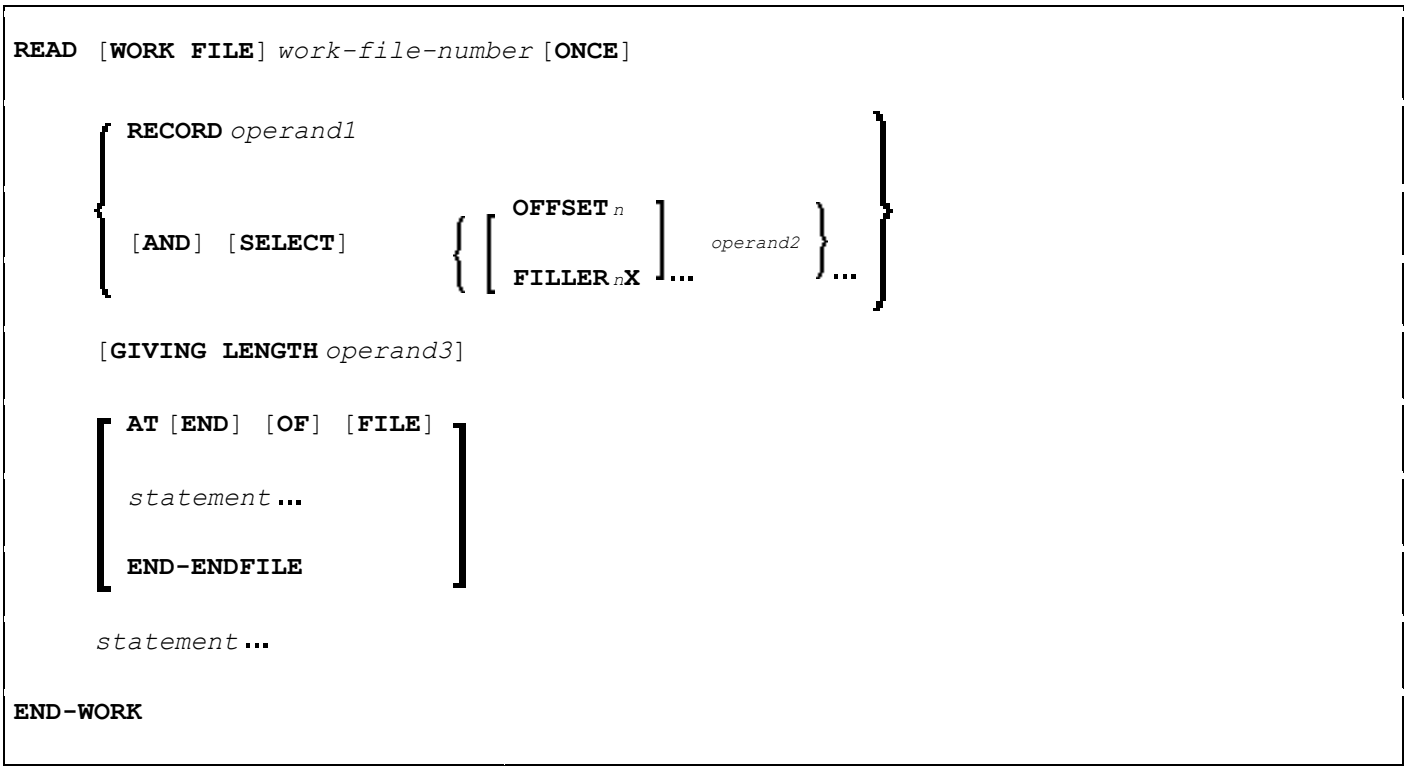
```
*
* WRITE WORK FILE  Procedo Batch(Ejemplos 1 PROGRAMA: EMPCARWR)
*
DEFINE DATA LOCAL
1 EMPLEADOS-VIEW VIEW OF EMPLEADOS
  2 DNI
  2 APELLIDO
  2 NOMBRE
  2 SALARIO
  2 CIUDAD
  2 FUNCIONES(10)
*
1 #COD-CARGO     (A3)
END-DEFINE
*
R1. READ EMPLEADOS-VIEW BY DNI        /* LEER TODOS LOS REGISTROS POR DNI
*
  DECIDE ON FIRST VALUE OF FUNCIONES(1)
    VALUE 'PROGRAMADOR SENIOR'
      MOVE 'PS' TO #COD-CARGO
    VALUE 'ANALISTA SENIOR'
      MOVE 'AS' TO #COD-CARGO
    VALUE 'JEFE PROYECTO'
      MOVE 'JP' TO #COD-CARGO
    NONE VALUE
      WRITE 'SC'       /* EMPLEADO SIN CARGO ASIGNADO
  END-DECIDE
*
  WRITE WORK FILE 01          /* CMWKF01 DEL JCL
        DNI
        APELLIDO
        NOMBRE
        SALARIO
        CIUDAD
        #COD-CARGO
*
END-READ
END
```


Autor: Yorman Guadarrama Martínez

READ WORK FILE:

La instrucción READ WOKR FILE es utilizada para leer registros desde un WORK FILIE en secuencia física.
A diferencia del WRITE WORK FILE, el READ WORK FILE inicia un procesamiento en bucle.

El resto de las condiciones son similares a las del WRITE WORK FILE, tomando en cuenta que al trabajar en bucle tiene una serie de opciones adicionales que no existen en el WRITE WORK FILE.




```
*
* READ WORK FILE  Procedo Batch(Ejemplos 1 PROGRAMA: EMPCARRE)
*
DEFINE DATA LOCAL
*
1 #REGISRO-EMPLEADOS-CARGO
  2 #DNI        (A10)
  2 #APELLIDO   (A20)
  2 #NOMBRE     (A15)
  2 #SALARIO    (N9.2)
  2 #CIUDAD     (A15)
  2 #COD-CARGO  (A3)
*
1 #CANTIDAD-REG (N3)
END-DEFINE
*
READ WORK FILE 01 RECORD #REGISRO-EMPLEADOS-CARGO           /* CMWKF01 DEL JCL
   DISPLAY NOTITLE
       #CIUDAD
       #COD-CARGO
       #DNI
       #APELLIDO
       #NOMBRE
       #SALARIO
END-WORK
END
```


Autor: Yorman Guadarrama Martínez

JCL: Job que ejecuta el programa natural con instrucción WRITE WORK FILE y READ WORK FILE.

```
//JOBEMP01 JOB,JOBEMP,CLASS=S,MSGCLASS=X,MSGLEVEL=(1,1)
//* ---------------------------------------- *
//* EJEMPLO: WRITE WORK FILE Y READ WORK FILE *
//* ---------------------------------------- *
//*
//PASO00   EXEC PGM=IEFBR14
//*
//DEL1     DD DSN=PRUEBA.EMPLEADOS.CARGOS.F160325,
//            DISP=(MOD,DELETE,DELETE),
//            SPACE=(TRK,2)
/*
//*
//* --- WRITE WORK FILE --- *
//*
//PASO01  EXEC NATBAT,BD=002,PARAM='INTENS=1',
//        NATCMD='DES.SISUTIL.LOGONNAT'
//CMWKF01  DD DSN=PRUEBA.EMPLEADOS.CARGOS.F160325,
//            UNIT=SYSDA,SPACE=(CYL,(10,1),RLSE),
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=80,BLKSIZE=8000,RECFM=FB,BUFNO=15)
//CMPRINT  DD SYSOUT=*
//CMPRT01  DD SYSOUT=*
//SYSIN    DD *
LOGON DESYGM
EMPCARWR
FIN
/*
//*
//* --- SORT (POR CUIDAD Y COD-CARGO)---
//*
//PASO02   EXEC PGM=SORT,REGION=1024K,PARM='INTENS=1',
//SORTIN    DD DSN=PRUEBA.EMPLEADOS.CARGOS.F160325,DISP=SHR
//SORTOUT   DD DSN=&&TEMP1,UNIT=SYSDA,SPACE=(CYL,(10,1),RLSE),
//             DISP=(NEW,PASS),
//             DCB=(LRECL=80,BLKSIZE=8000,RECFM=FB,BUFNO=15)
//SYSOUT    DD SYSOUT=*
//SYSIN     DD *
SORT FIELDS=(57,15,CH,A,72,3,CH,A,1,10,CH,A)
/*
//*
//* --- READ WORK FILE --- *
//*
//PASO03  EXEC NATBAT,BD=002,PARAM='INTENS=1',
//        NATCMD='DES.SISUTIL.LOGONNAT'
//CMWKF01  DD DSN=&&TEMP1,DISP=SHR
//CMPRINT  DD SYSOUT=*
//CMPRT01  DD SYSOUT=*
//SYSIN    DD *
LOGON DESYGM
EMPCARRE
FIN
/*
```


Autor: Yorman Guadarrama Martínez

EDITOR PARA DESARROLLAR OBJETOS

MAIN MENU:

```
11:34:16                    *****  NATURAL  *****                    2016-07-20
User DEYGUA                     - Main Menu -                      Library DES002


                    Function

                 X Development Functions
                 _ Development Environment Settings
                 _ Maintenance and Transfer Utilities
                 _ Debugging and Monitoring Utilities
                 _ Example Libraries
                 _ Other Products
                 _ Help
                 _ Exit Natural Session




Logon accepted to library DES002.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit                                                 Canc
```

DEVELOPMENT FUNCTIONS:

```
11:39:32                    *****  NATURAL  *****                    2016-07-20
User DEYGUA               - Development Functions -                 Library DES002


                           Code Function

                            C    Create Object
                            E    Edit Object
                            R    Rename Object
                            D    Delete Object
                            X    Execute Program
                            L    List Object(s)
                            S    List Subroutines Used
                            ?    Help
                            .    Exit

                     Code ____     Type ____
                                   Name ____________________________


Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Menu  Exit                                                 Canc
```


Autor: Yorman Guadarrama Martínez

CREAR UN OBJETO NATURAL:

```
11:39:32                    *****  NATURAL  *****                    2016-07-20
User DEYGUA                - Development Functions -                 Library DES002
                                                  +---------OBJECT TYPE----------+
                                                  |        M  Map                |
                           Code Function          |        G  Global             |
                                                  |        L  Local              |
                            C    Create Object|   |        A  Parameter          |
                            E    Edit Object  |   |        P  Program            |
                            R    Rename Object|   |        N  Subprogram         |
                            D    Delete Object|   |        S  Subroutine         |
                            X    Execute Progr|   |        H  Helproutine        |
                            L    List Object(s|   |        C  Copycode           |
                            S    List Subrouti|   |        T  Text               |
                            ¿    Help         |   |        .  Exit               |
                            .    Exit         |   |   Type P                     |
                                              |   |   Name PROG0001              |
                      Code .. C    Type .. ?    +------------------------------+
                                   Name _____________________


Command =
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10—PF11—PF12---
       Help  Menu  Exit                                                  Canc
```

EDITAR UN OBJETO NATURAL:

```
11:39:32                    *****  NATURAL  *****                    2016-07-20
User DEYGUA                - Development Functions -                 Library DES002


                           Code Function

                            C    Create Object
                            E    Edit Object
                            R    Rename Object
                            D    Delete Object
                            X    Execute Program
                            L    List Object(s)
                            S    List Subroutines Used
                            ?    Help
                            .    Exit

                     Code .. E    Type .. P
                                  Name PROG0001_


Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Menu  Exit                                                  Canc
```


Autor: Yorman Guadarrama Martínez

```
>     línea-comandos...                        > + Program     PROG0001   Lib DES002
All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
   0010 DEFINE DATA LOCAL
   0020 *
   0030 01 #REGISRO-EMPLEADOS-CARGO
   0040   02 DNI
   0050   02 APELLIDO
   0060   02 NOMBRE
   0070   02 SALARIO
   0080   02 CIUDAD
   0090 END-DEFINE
   0100 *
   0110 READ EMPLEADOS-VIEW BY APELLIDO STARTING FROM 'MORENO'
   0120 *
   0130    DISPLAY NOTITLE (ES=OFF IS=ON ZP=ON AL=15)
   0140         DNI
   0150         APELLIDO
   0160         NOMBRE
   0170         CIUDAD
   0180    END-READ
   0190 *
   0200 END
        ....+....1....+....2....+....3....+....4....+....5....+... S 15   L 1
```

COMANDOS DE LA LÍNEA DE COMANDOS:

Se ejecutan en la línea de comandos de la esquina superior izquierda del editor. Alguno de ellos también puede ser ejecutado en la línea de comandos de la pantalla **"Development Functions"**.

- **SAVE *nombre-programa*:** Salva o guarda un programa fuente.
- **STOW:** Salva y compila un programa fuente o un programa objeto Natural. Verifica que el programa no tenga errors y lo compila.
- **RUN:** Ejecuta un programa fuente.
- **Execute *nombre-programa*:** Ejecuta el programa objeto (compilado). Este comando es opcional y puede escribirse el nombre del programa directamente.
- **STRUCT:** Permite tener un código fuente estructurado.
- **SCAN:** Este comando tiene múltiples funciones en Natural, pero desde esta línea de comandos permite la apertura de un menú para realizar búsquedas y remplazos de una o varias cadena de caracteres.
- **CHECK:** Comprueba errores de sintaxis.
- **CLEAR:** Limpia el código fuente del editor.
- **READ *nombre-programa*:** Carga otro código fuente en el editor (cargar en memoria).
- **DELETE *nombre-programa*:** Borra un programa objeto.
- **SCRATCH *nombre-programa*:** NO SE RECOMIENDA SU USO. En lugar de este usar el DELETE.
- ***:** muestra el ultimo commando ejecutado.
- **N:** Renumera las líneas de código en el editor.
- **CANCEL:** Sale del editor sin salvar los últimos cambios.
- **LET:** Deshace todas las líneas de código modificadas antes de presionar el ultimo [Enter].

- **DX, DY:** Borra la línea de código marcada con X o Y.
- **DX-Y:** Borra el bloque de líneas de código que estén desde la X hasta la Y.
- **EX, EY:** Borra todas las líneas de código del editor excepto la marcada con X o Y.
- **EX-Y:** Borra todas las líneas de código del editor excepto las que estén desde la X hasta la Y.
- **PF7, -:** Retrocede una página.
- **PF8, +:** Avanza una página.
- **-H:** Retrocede una media página.
- **+H:** Avanza una media página.
- **T, --:** Retrocede al principio del programa.
- **B, ++:** Avanza al final del programa.
- **X, Y:** Se posiciona en la línea de código marcada con una X o Y.
- **RESET:** Desmarca las líneas X o Y.

COMANDOS DE LAS LÍNEAS DEL CÓDIGO FUENTE:

Se ejecutan en la primera posición de cada línea del código fuente.

- **.X, .Y:** Marca la línea con X o Y.
- **.P:** Posiciona la línea al principio.
- **.MX, .MY:** Mueve la línea marcada con X o Y a la línea donde se ejecuta el comando.
- **.CX, .CY:** Copia la línea marcada con X o Y a la línea donde se ejecuta el comando.
- **.D:** Borra la línea marcada con X o Y a la línea donde se ejecuta el comando.
- **.MX-Y:** Mueve el bloque de líneas desde la X a Y, a la línea donde se ejecuta el comando.
- **.CX-Y:** Copia el bloque de líneas desde la X a Y, a la línea donde se ejecuta el comando.
- **.I:** Inserta una línea vacía.
- **.J:** Une la línea donde se ejecuta el comando con la siguiente.
- **.L:** Deshace las últimas modificaciones de la línea antes de presionar el [ENTER].
- **.S:** Corta la línea donde se ejecuta el comando justo donde se posicione el cursor.
- **.C, .D, .I:** Pueden también usar un parámetro "*n*" para ejecutar su función a "*n*" líneas.
- **.I(*nombre-Objeto*):** Insertar un objetos Natural (LDA,GDA,PDA,DDM,Programa,etc..) solo si este está definido en la misma librería o en la STEPLIB, por defecto la STEPLIB es la librería SYSTEM.

CREAR UN OBJETO NATURAL TIPO LDA:

```
11:39:32                    *****  NATURAL  *****                    2016-07-20
User DEYGUA                - Development Functions -                 Library DES002
                                                  +---------OBJECT TYPE----------+
                                                  |        M  Map                |
                           Code Function          |        G  Global             |
                                                  |        L  Local              |
                            C    Create Object|   |        A  Parameter          |
                            E    Edit Object  |   |        P  Program            |
                            R    Rename Object|   |        N  Subprogram         |
                            D    Delete Object|   |        S  Subroutine         |
                            X    Execute Progr|   |        H  Helproutine        |
                            L    List Object(s|   |        C  Copycode           |
                            S    List Subrouti|   |        T  Text               |
                            ¿    Help         |   |        .  Exit               |
                            .    Exit         |   |   Type L                     |
                                              |   |   Name LDA0001               |
                      Code .. C    Type .. ?    +------------------------------+
                                   Name _____________________


Command =
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10—PF11—PF12---
       Help  Menu  Exit                                                  Canc
```

Con el commando ".V(*nombre-DDM*)" se inserta la DDM del fichero Adabas:

```
  Local     LDA0001     Library DES002                              DBID  10 FNR  32
  Command                                                                 > +
  I T L  Name                              F Length     Miscellaneous
  All -- ------------------------------- - ---------- --------------------------
    *    LDA for new application
      .V(nombre-DDM)









-- Current Source Size: 625  Free: 61408 ------------------------- S 12   L 1
```

Este comando inserta todos los campos de la DDM y se seleccionan solo los campos que sean requeridos en la LDA.

Autor: Yorman Guadarrama Martínez

```
Local     LDA0001     Library DES002                          DBID  10 FNR  32
Command                                                              > +
I T L  Name                             F Length     Miscellaneous
All -- -------------------------------- - ---------- --------------------------
  *    LDA for new application
     1 INGRESO                          A         20 (1:3,1:5) INIT ALL<'0'>
     1 PERSONA
     2 SEXO                             A          6
     2 EDAD                             N          3
     1 NOMBRE                           A         24
  R  1 NOMBRE                                        /* REDEF. BEGIN : NOMBRE
     2 NOMBRE                           A         10
     2 INICIAL                          A          2
     2 APELLIDO                         A         10
  C  1 EURO                             A          5 CONST<'€'>
  V  1 FINANZA-VIEW                                  FINANZA
     2 DNI                              A         10
  P  2 CREDITO-PPAL                                  (1:5) /* PERIODIC GROUP
     3 TARJETA-CREDITO                  A         18 (EM=XXX.XXX.XXX.XXX.XXX.XXX)
     3 LIMITE-CREDITO                   N        4.0
     3 BALANCE-ACTUAL                   N        4.0
-- Current Source Size: 625  Free: 61408 -------------------------- S 12   L 1
```


Autor: Yorman Guadarrama Martínez

MAPAS

CONCEPTO E IMPLEMENTACIÓN:

Este tipo de objeto es utilizado para diseñar pantallas dinámicas. Un MAPA es llamado o ejecutado desde un programa NATURAL usando la instrucción **INPUT USING MAP *nombre-mapa*.**

El uso de MAPAS ofrece la posibilidad de:

- Separar la lógica del programa NATURAL con su entrada y salida de datos.
- Modificar el diseño del MAPA si realizar cambios en el programa NATURAL.
- Mantener una programación estructurada.

TIPOS DE MAPAS:

- **INPUT MAP:** Para pantallas con campos de entrada, salida y modificables.
- **OUTPUT MAP:** Para generar reportes por pantalla. (solo campos de salida).
- **HELP MAP:** Son iguales a los INPUT MAP, pero adicionalmente pueden ser usados para mostrar mensajes de ayuda.

```
INPUT [WINDOW='window-name'] [WITH-TEXT-option]

  [MARK-option]

  [ALARM-option]

  [USING] MAP map-name [NO ERASE]

  operand1 ...

  NO PARAMETER
```


Autor: Yorman Guadarrama Martínez

```
*
* INPUT USING MAP (Ejemplo programa PROG0001)
*
DEFINE DATA LOCAL
1 EMPLEADOS-VIEW VIEW OF EMPLEADOS
  2 DNI
  2 APELLIDO
  2 NOMBRE
  2 SALARIO
  2 CIUDAD
*
1 #DNI        (A10)
1 #APELLIDO   (A20)
1 #NOMBRE     (A15)
1 #SALARIO    (N9.2)
1 #CIUDAD     (A15)
1 #CONFIRMAR  (A1)
1 #ACTUALIZAR (A1)
1 #NOMBPROG   (A8)
END-DEFINE
*
MOVE *PROGRAM TO #NOMBPROG     /* PROG0001
*
R1. REPEAT
   INPURT USING MAP MAPA0001 NO ERASE
*
   F1. FIND EMPLEADOS-VIEW WITH DNI = #DNI    /* SELECCIONAR REG.
      IF NO RECORD FOUND
         REINPUT 'DNI DEL EMPLEADO NO EXISTE INTENTE DE NUEVO' MARK 1 ALARM
                                                          /* MARK *#DNI ALARM
      END-NOREC
      IF *NUMBER = 1
         MOVE *ISN(F1.) TO #ISN
         /*
         #DNI      := EMPLEADOS-VIEW.DNI
         #APELLIDO := EMPLEADOS-VIEW.APELLIDO
         #NOMBRE   := EMPLEADOS-VIEW.NOMBRE
         #SALARIO  := EMPLEADOS-VIEW.SALARIO
         #CIUDAD   := EMPLEADOS-VIEW.CIUDAD
      ELSE
         REINPUT 'DNI DEL EMPLEADO ESTA DUPLICADO' MARK *#DNI ALARM
      END-IF
   END-FIND
*
   IF #ACTUALIZAR <> 'S' AND #ACTUALIZAR <> 'N'
      REINPUT 'CONFIRMAR LA ACTUALIZACION....' MARK *#ACTUALIZAR ALARM
   END-IF
*
   /* IF (VALIDACIONES DE LOS CAMPOS DEL MAPA)...
   /*     REINPUT 'MENSAJE DE VALIDACIONES...' MARK *campo-mapa ALARM
   /* END-IF
*
   IF #ACTUALIZAR = 'S'
      G1. GET EMPLEADOS-VIEW #ISN             /* LEER Y BLOQUEAR REG.
*
      EMPLEADOS-VIEW.DNI      := #DNI
      EMPLEADOS-VIEW.APELLIDO := #APELLIDO
      EMPLEADOS-VIEW.NOMBRE   := #NOMBRE
      EMPLEADOS-VIEW.SALARIO  := #SALARIO
      EMPLEADOS-VIEW.CIUDAD   := #CIUDAD
*
      UPDATE (G1.)                            /* ACTUALIZAR REG.
      END TRANSACTION                         /* CONSOLIDAR ACTUALIZACION.
   END-IF
*
```


Autor: Yorman Guadarrama Martínez

```
   IF #CONFIRMAR <> 'S' AND #CONFIRMAR <> 'N'
      REINPUT 'CONTINUAR MODIFICANDO EMPLEADOS (S/N):' MARK *#CONFIRMAR ALARM
   END-IF
   IF #CONFIRMAR = 'N'
      ESCAPE BOTTOM(R1.)
   END-IF
*
   RESET #DNI #APELLIDO #NOMBRE #SALARIO #CIUDAD #CONFIRMAR #ACTUALIZAR
END-REPEAT
END
```

```
PROG0001                                                              15/03/16
MAPA0001               ACTUALIZACION DE DATOS DE EMPLEADOS            14:35:10
                       ------------ -- ----- -- ---------

       Indique DNI del Empleado: 356823947T

       ------------------------------------------------------------------------

       Apellido...........:  LOPEZ

       Nombre.............:  MARTA

       Salario............:  1325.50

       Ciudad.............:  MADRID

       ------------------------------------------------------------------------

       Desea modificar los datos del empleado (S/N):  S

       Continuar Modificando Empleados (S/N):  N
```


Autor: Yorman Guadarrama Martínez

EDITOR PARA DISEÑAR MAPAS

```
14:53:29                ***** NATURAL MAP EDITOR *****              2016-07-20
User DEYGUA                        - Edit Map -                   Library DES002

                    Code    Function
                    ----    --------------------------------
                      D     Field and Variable Definitions
                      E     Edit Map
                      I     Initialize new Map
                      H     Initialize a new Help Map
                      M     Maintenance of Profiles & Devices
                      S     Save Map
                      T     Test Map
                      W     Stow Map
                      ?     Help
                      .     Exit



              Code I___      Name MAPA0001_       Profile SYSPROF_

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Test  Edit
```

CON LA OPCIÓN "I" SE EDITA EL MAPA:

```
10:41:16               Define Map Settings for MAP                  2016-07-20

 Delimiters           Format                        Context
 -----------------    ---------------------------   --------------------------
 Cls Att CD  Del      Page Size ..... 23            Device Check .... ________
  T   D      BLANK    Line Size ..... 79            WRITE Statement
  T   I      ?        Column Shift ... 0 (0/1)      INPUT Statement   X
  A   D      _        Layout ........ ________      Help ____________________
  A   I      )         dynamic ..... N (Y/N)         as field default N (Y/N)
  A   N      ¬        Zero Print ..... N (Y/N)
  M   D      &        Case Default ... UC (UC/LC)
  M   I      :        Manual Skip .... N (Y/N)      Automatic Rule Rank  1
  O   D      +        Decimal Char ... .            Profile Name .... SYSPROF
  O   I      (        Standard Keys .. N (Y/N)
  D   D      $        Justification .. L (L/R)      Filler Characters
  D   I      /        Print Mode ..... __           ------------------------
                                                    Optional, Partial ....
                      Control Var .... ________     Required, Partial ....
                                                    Optional, Complete ...
                                                    Required, Complete ...

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                  Let
```

En esta pantalla se configura el perfil del mapa, donde se establece el acho/largo, asignar un nombre mapa diseñado, la supresión de ceros, mayúsculas/minúsculas, carácter decimal, justificar derecha/izquierda, variable de control, tipo de mapa INPUT/WRITE, Helprutina y reglas de procesamiento.

Autor: Yorman Guadarrama Martínez

CON LA OPCIÓN "E" SE EDITA EL MAPA:

```
14:53:29                ***** NATURAL MAP EDITOR *****              2016-07-20
User DEYGUA                       - Edit Map -                    Library DES002

                   Code    Function
                   ----    --------------------------------
                     D     Field and Variable Definitions
                     E     Edit Map
                     I     Initialize new Map
                     H     Initialize a new Help Map
                     M     Maintenance of Profiles & Devices
                     S     Save Map
                     T     Test Map
                     W     Stow Map
                     ?     Help
                     .     Exit



             Code E___      Name MAPA0001_       Profile SYSPROF_

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Test  Edit
```

```
Editing completed successfully.
Ob _                                    Ob D CLS ATR  DEL    CLS ATR  DEL
.                                       .      T  D    Blnk    T  I    ?
.                                       .      A  D    _       A  I    )
.                                       .      A  N    ¬       M  D    &
.                                       .      M  I    :       O  D    +
.                                       .      O  I    (
.                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
+XXXXXXXX                                                                +XXXXXXXX
+XXXXXXXX               ?ACTUALIZACION?DE?DATOS?DE?EMPLEADOS             +XXXXXXXX
                        ?-------------?--?-----?--?---------


      ?Indique?DNI?del?Empleado: :XXXXXXXXXX

      ?-------------------------------------------------------------------------

       Apellido...........: &XXXXXXXXXXXXXXXXXXXX

       Nombre.............: &XXXXXXXXXXXXXXX

       Salario............: &999999999.99
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```


Autor: Yorman Guadarrama Martínez

CON LA TECLA "PF9" SE MUESTRA LA PANTALLA COMPLETA:

```
+XXXXXXXX                                                              +XXXXXXXX
+XXXXXXXX              ?ACTUALIZACION?DE?DATOS?DE?EMPLEADOS            +XXXXXXXX
                       ?-------------?--?-----?--?---------


      ?Indique?DNI?del?Empleado: :XXXXXXXXXX

      ?-------------------------------------------------------------------------

       Apellido...........: &XXXXXXXXXXXXXXXXXXXX

       Nombre.............: &XXXXXXXXXXXXXXX

       Salario............: &999999999.99

       Ciudad.............: &XXXXXXXXXXXXXXX

      ?-------------------------------------------------------------------------

      ?Desea?modificar?los?datos?del?empleado?(S/N): &X

       Continuar?Modificando?Empleados?(S/N): &X


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```

CON LA TECLA "PF4" SE PRUEBA EL DISEÑO DEL MAPA:

```
                                                                       15/03/16
MAPA0001               ACTUALIZACION DE DATOS DE EMPLEADOS             14:35:10
                       ------------ -- ----- -- ---------

       Indique DNI del Empleado:

       -------------------------------------------------------------------------

       Apellido...........:

       Nombre.............:

       Salario............:

       Ciudad.............:

       -------------------------------------------------------------------------

       Desea modificar los datos del empleado (S/N):

       Continuar Modificando Empleados (S/N):
```


Autor: Yorman Guadarrama Martínez

LA OPCIÓN "Ob" SE UTILIZA PARA REFERENCIAR LOS CAMPOS USADOS EN EL MAPA:

P(Programas), V(DDM), L(LDA), G(GDA), A(PDA) N(Subprograma), S(Subrutina), M(Mapa), H(Helprutina).

```
 Ob P PROG0001                              Ob D  CLS ATR DEL     CLS ATR DEL
  1 #DNI                      A10            .      T  D    Blnk    T  I    ?
  2 #APELLIDO                 A20            .      A  D    _       A  I    )
  3 #NOMBRE                   A15            .      A  N    ¬       M  D    &
  4 #SALARIO                  N9.2           .      M  I    :       O  D    +
  5 #CIUDAD                   A15            .      O  I    (
  6 #CONFIRMAR                A1             .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+-
+XXXXXXXX                                                                  +*DATE
+*PROGRAM               ?ACTUALIZACION?DE?DATOS?DE?EMPLEADOS                +*TIMX
                        ?-------------?--?-----?--?---------


      ?Indique?DNI?del?Empleado: :1

      ?-------------------------------------------------------------------

       Apellido...........: &2

       Nombre.............: &3

       Salario............: &4
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

Se presiona [INTRO].

```
  Ob _                                      Ob D CLS ATR DEL     CLS ATR DEL
   .                                         .     T  D    Blnk    T  I    ?
   .                                         .     A  D    _       A  I    )
   .                                         .     A  N    ¬       M  D    &
   .                                         .     M  I    :       O  D    +
   .                                         .     O  I    (
   .                                         .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
+XXXXXXXX                                                                  +XXXXXXXX
+XXXXXXXX               ?ACTUALIZACION?DE?DATOS?DE?EMPLEADOS               +XXXXXXXX
                        ?-------------?--?-----?--?---------


      ?Indique?DNI?del?Empleado: :XXXXXXXXXX

      ?-------------------------------------------------------------------

       Apellido...........: &XXXXXXXXXXXXXXXXXXX

       Nombre.............: &XXXXXXXXXXXXXXX

       Salario............: &999999999.99
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```


Autor: Yorman Guadarrama Martínez

LA OPCIÓN "Ob" TAMBIÉN SE UTILIZA PARA DESLAZAR LA LISTA DE CAMPOS:

1. (+) Se desplaza una página hacia abajo.
2. (++) Se desplaza al final.
3. (-) Se desplaza al principio.
4. (--) Se desplaza una página hacia arriba.
5. (+*n*) Se desplaza *n* líneas hacia abajo.
6. (-*n*) Se desplaza *n* líneas hacia arriba.

```
 Ob + 4                                    Ob D  CLS ATR DEL     CLS ATR DEL
  1 #CIUDAD                  A15            .      T  D    Blnk    T  I    ?
  2 #CONFIRMAR               A1             .      A  D    _       A  I    )
  3 #ACTUALIZAR              A1             .      A  N    ¬       M  D    &
  4 #NOMBPROG                A8             .      M  I    :       O  D    +
  .                                         .      O  I    (
  .                                         .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+-

       Ciudad.............: &1

      ?-------------------------------------------------------------------------

      ?Desea?modificar?los?datos?del?empleado?(S/N): &3

       Continuar?Modificando?Empleados?(S/N): &2




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```


Autor: Yorman Guadarrama Martínez

DEFINICIÓN DE ATRIBUTOS DE CAMPOS:

CLS (Clase de campo).

1. T: Titilo.
2. A: Solo de entrada (Solo para ingresar datos).
3. M: Entrada y salida (Modificable).
4. O: Solo salida (Solo para mostrar datos).

ATR (Atributos).

1. D: Mostrar sin atributos adicionales.
2. N: No visible.
3. I: Mostrar intensivo.

DEL (Delimitador).

El prefijo utilizado para cada combinación de CLS+ATR.

```
Ob _                                    Ob D CLS ATR  DEL    CLS ATR  DEL
.                                       .      T  D    Blnk    T  I    ?
.                                       .      A  D    _       A  I    )
.                                       .      A  N    ¬       M  D    &
.                                       .      M  I    :       O  D    +
.                                       .      O  I    (
.                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
+XXXXXXXX                                                                +XXXXXXXX
+XXXXXXXX              ?ACTUALIZACION?DE?DATOS?DE?EMPLEADOS              +XXXXXXXX
                       ?-------------?--?-----?--?---------


      ?Indique?DNI?del?Empleado: :XXXXXXXXXX

      ?-----------------------------------------------------------------------

       Apellido...........: &XXXXXXXXXXXXXXXXXXXX

       Nombre.............: &XXXXXXXXXXXXXXX

       Salario............: &999999999.99
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```


Autor: Yorman Guadarrama Martínez

COMANDOS DE LÍNEA MÁS UTILIZADOS:

1. "..M" Mueve la línea donde fue introducido el comando debajo de la posición del cursor.
2. "..C" Centra el literal de la línea donde fue introducido el comando.
3. "..R" Copia la línea donde fue introducido el comando a la posición del cursor.
4. "..S" Divide la línea donde fue introducido el comando en la posición del cursor.
5. "..J" Une la línea donde fue introducido el comando, con la siguiente línea.
6. "..I" Inserta una línea donde fue introducido el comando. La última línea de la pantalla es borrada.
7. "..D" Elimina la línea donde fue introducido el comando.
8. Si se agrega el número *"n"* al comando, este ejecutará la acción *"n"* líneas incluyendo la línea donde fue introducido el comando. Ejemplo: *"..Cn" o "..Dn".*

```
  Ob _                                    Ob D CLS ATR DEL     CLS ATR DEL
   .                                       .     T  D    Blnk    T  I    ?
   .                                       .     A  D    _       A  I    )
   .                                       .     A  N    ¬       M  D    &
   .                                       .     M  I    :       O  D    +
   .                                       .     O  I    (
   .                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
+XXXXXXXX                                                                 +XXXXXXXX
+XXXXXXXX    ..D2       ?ACTUALIZACION?DE?DATOS?DE?EMPLEADOS              +XXXXXXXX
                        ?-------------?--?-----?--?---------


      ?Indique?DNI?del?Empleado: :XXXXXXXXXX

 ..C3 ?-------------------------------------------------------------------------

       Apellido...........: &XXXXXXXXXXXXXXXXXXXX

       Nombre.............: &XXXXXXXXXXXXXXX

       Salario............: &999999999.99
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```


Autor: Yorman Guadarrama Martínez

COMANDOS DE CAMPO MÁS UTILIZADOS:

1. ".M" Mueve el campo donde fue introducido el comando a la posición del cursor.
2. ".C" Centra el campo donde fue introducido el comando entre dos campos adyacentes en la misma línea.
3. ".R" Copia el campo donde fue introducido el comando a la posición del cursor.
4. ".S" Divide la línea justo en el campo donde fue introducido el comando.
5. ".J" Une el campo donde fue introducido el comando con los campos de la siguiente línea, hasta la posición del cursor.
6. ".D" Elimina el campo donde fue introducido el comando.
7. Estos comandos también pueden ser utilizados para procesar rangos de campos, delimitando el campo inicio y fin.

```
  Ob _                                    Ob D CLS ATR DEL     CLS ATR DEL
   .                                       .     T  D    Blnk    T  I    ?
   .                                       .     A  D    _       A  I    )
   .                                       .     A  N    ¬       M  D    &
   .                                       .     M  I    :       O  D    +
   .                                       .     O  I    (
   .                                       .
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
+XXXXXXXX                                                                 +XXXXXXXX
+XXXXXXXX              .CCTUALIZACION?DE?DATOS?DE.CMPLEADOS               +XXXXXXXX
                       ?-------------?--?-----?--?---------


      ?Indique?DNI?del?Empleado: :XXXXXXXXXX

      .D-------------------------------------------------------------------------

       Apellido...........: &XXXXXXXXXXXXXXXXXXXX

       Nombre.............: .MXXXXXXXXXXXXXX

       Salario............: &999999999.99
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```


Autor: Yorman Guadarrama Martínez

OTROS COMANDOS DE CAMPO:

1. ".E" Edita las especificaciones extendidas del campo donde fue introducido el comando (***Área de edición extendida***).
2. ".P" Edita el código de las ***reglas de procesamiento*** asociadas al campo donde fue introducido el comando.
3. ".A" Edita las especificaciones de un campo ***tipo arreglo***, donde fue introducido el comando.

Ejemplo comando .E

```
Fld #DNI                                                          Fmt A10
-------------------------------------------------------------------------------
AD= MIT_______     ZP=         SG=        HE= _____________________   Rls 0
AL= ______         CD= __      CV= _____________________________   Mod User
PM= __   DF=       BX= ____    DY= _____________________________
EM= ____________________________ SB= _____________________________

001   --010---+----+----+---030---+----+----+---+050---+----+----+---+070---+----
+XXXXXXXX                                                              +XXXXXXXX
+XXXXXXXX               ?ACTUALIZACION?DE?DATOS?DE?EMPLEADOS           +XXXXXXXX
                        ?-------------?--?-----?--?---------


      ?Indique?DNI?del?Empleado: .EXXXXXXXXX

      ?-----------------------------------------------------------------------

       Apellido...........: &XXXXXXXXXXXXXXXXXXXX

       Nombre.............: &XXXXXXXXXXXXXXX

       Salario............: &999999999.99
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

Es un área de edición extendida del campo que permite asignar atributos adicionales a este con el máximo detalle. Los parámetros y sus opciones más utilizados son:

1. AD: Además de los atributos de campo anteriormente mencionado, existen otros que permiten mostrar el campo subrayado, justificar derecha/izquierda, justificar los ceros a la derecha, acepta mayúsculas y minúsculas, cursiva/itálica, parpadeando, con protección temporal (en combinación con el parámetro CV).
2. ZP: Para suprimir los ceros no significativos en un campo de salida (ON/OFF).
3. SG: Para asignar un signo solo en campos numéricos (ON/OFF).
4. HE: Para asignar una Helprutina al campo.
5. CV: Se define como variable de control y es utilizada para modificar los atributos del campo predefinidos al diseñar el mapa, durante la ejecución del programa que lo llama.
6. DF: Para formatos de campo tipo fecha (Date).
7. EM: Permite diseñar una máscara para el campo ("X"-Alfanumérico, "9/Z" – Numérico, "H"-Hexadecimal o Binario), adicional proporciona mascara específicas para campos tipo Date y Time (HH:II:SS).

Autor: Yorman Guadarrama Martínez

Ejemplo comando .P

```
Fld #DNI                                                          Fmt A10
------------------------------------------------------------------------------
AD= MIT_______     ZP=         SG=        HE= _____________________   Rls 0
AL= _____          CD= __      CV= _____________________________   Mod User
PM= __  DF=        BX= ____    DY= _____________________________
EM= _____________________ SB= _____________________________

001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
+XXXXXXXX                                                          +XXXXXXXX
+XXXXXXXX              ?ACTUALIZACION?DE?DATOS?DE?EMPLEADOS        +XXXXXXXX
                       ?-------------?--?-----?--?---------


      ?Indique?DNI?del?Empleado: &XXXXXXXXXX

      ?-------------------------------------------------------------------

       Apellido...........: .PXXXXXXXXXXXXXXXXXX

       Nombre.............: &XXXXXXXXXXXXXX

       Salario............: &999999999.99
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
```

```
  Variables used in current map                                          MOD
  #DNI(A10)                                                               U
  #NOMBRE(A15)                                                            U
  #CIUDAD(A15)                                                            U
  #SALARIO(N9.2)                                                          U
  #CONFIRMAR(A1)                                                          U
  *DATE                                                                   S

  Rule _____________________________ Field PROG0001.#APELLIDO
  >                                   > + Rank 0     S 1   L 1    Struct Mode
  ALL   ....+....10...+....20...+....30...+....40...+....50...+....60...+....70..
    0010 *
    0020 IF & = '                    '
    0030  REINPUT 'DEBE INDICAR EL APELLIDO DEL EMPLEADO'
    0040 END-IF
    0050 *
    0060
    0070
    0080
    0090
    0100
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Mset  Exit  Test        --    -     +     Full  Sc=         Let
```

El comando de campo **.P** adicionalmente puede ser utilizado con los siguientes parámetros:

- **.P*rr*:** Se obtiene la regla ***Prr,*** d*onde* ***rr*** representa el nivel de prioridad de ejecución de la regla ***(Rank)***. Hay disponibles 100 niveles (P0 – P99).
- **.P*:** Se obtiene una lista de todas las reglas definidas en el campo para seleccionar una de estas.

En la línea de comandos del editor de reglas de procesamiento **(> ... >)** pueden utilizarse los siguientes comandos directos de regla:

Autor: Yorman Guadarrama Martínez

- **P*rr*:** Seleccionar la regla ***rr*** asociada al campo, donde ***rr*** representa la prioridad de ejecución de la regla ***(Rank)***. Hay disponible 100 niveles (**P0–P99)**.
- **P*:** Seleccionar una de todas las reglas listadas en el menú de selección.
- **P=*rr*:** Modificar el nivel de prioridad de la regla.

INDICADOR MOD:

En cualquiera de las pantallas del editor de mapas, el parámetro **MOD** muestra el origen de las variables usadas en él:

- **(U)** Variables de Usuarios: Definidas por el usuario en el programa que llama al mapa.
- **(S)** Variables de Sistemas: Son las variables de sistemas utilizadas en el mapa. Pueden estar definidas en el programa o directamente en el mapa.
- **(D)** Variables de Vista: son las variables definidas en la vista del programa, que a su vez es una definición lógica de la DDM EMPLEADOS.

Ejemplo comando .A en un arreglo con formato y longitud (A2) de una dimensión y una Occ.

```
Name #001                          Top Dim 1__________ 1__________ 1__________
 ------------------------------------------------------------------------------
 Dimensions                      Occurrences    Starting from     Spacing
 0 . Index vertical              1__            ____________      0   Lines
 0 . Index horizontal            1__            ____________      1   Columns
 0 . Index (h/v) V               1__            ____________      0   Cls/Ls

 001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----


    Introduzca código asociado de empresa: .AX









  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Mset  Exit             --    -     +          <     >     Let
```

Este ejemplo muestra el área de definición de un arreglo una vez ejecutado el comando de campo **.A**, donde se dispone de las siguientes especificaciones:

- **Top Dim:** Determina las dimensiones que puede tener un arreglo (Una, dos o tres).
- **Dimensions:** Determina el orden en el cual las dimensiones del arreglo son recorridas usando el índice de estas.
  1. **Occurrences:** Número de ocurrencias de cada dimensión.

Autor: Yorman Guadarrama Martínez

2. **Starting from:** El valor inicial del índice de cada dimensión. Este número puede ser una constante o una variable.
3. **Spacing:** El espacio en líneas o columnas entre cada índice de la dimensión.

Ejemplo 1:

```
Name #EJ1                          Top Dim 10_________ 1__________ 1__________
  ------------------------------------------------------------------------------
  Dimensions                       Occurrences    Starting from     Spacing
  1 . Index vertical               10_            ________          1   Lines
  0 . Index horizontal             1__            ________          1   Columns
  0 . Index (h/v) V                1__            ________          0   Cls/Ls
```

Arreglo con una dimensión vertical de 10 Occ y un espacios en blanco entre cada Occ.

Ejemplo 2:

```
  Name #EJ2                          Top Dim 10_________ 1__________ 1__________
  ------------------------------------------------------------------------------
  Dimensions                       Occurrences    Starting from     Spacing
  0 . Index vertical               1__            ________          0   Lines
  1 . Index horizontal             10_            ________          1   Columns
  0 . Index (h/v) V                1__            ________          0   Cls/Ls
```

Arreglo con una dimensión horizontal de 10 Occ y 1 espacios en blanco entre cada Occ.

Ejemplo 3:

```
  Name #EJ3                          Top Dim 10_________ 5__________ 1__________
  ------------------------------------------------------------------------------
  Dimensions                       Occurrences    Starting from     Spacing
  1 . Index vertical               10_            ________          1   Lines
  2 . Index horizontal             5__            ________          2   Columns
  0 . Index (h/v) V                1__            ________          0   Cls/Ls
```

Arreglo con dos dimensiones. Vertical de 10 Occ y horizontal de 5 Occ. Un espacios en blanco entre cada Occ vertical y 2 espacios en blanco entre cada Occ horizontal. El arreglo será recorrido en el orden siguiente, primero dimensión-vertical y luego dimensión-horizontal.

Ejemplo 4:

```
  Name #EJ4                          Top Dim 5_________ 10_________ 1__________
  ------------------------------------------------------------------------------
  Dimensions                       Occurrences    Starting from     Spacing
  2 . Index vertical               10_            ________          1   Lines
  1 . Index horizontal             5__            ________          2   Columns
  0 . Index (h/v) V                1__            ________          0   Cls/Ls
```

Exactamente igual al ejemplo 4, excepto que el arreglo será recorrido en el orden siguiente, primero dimensión-horizontal y luego dimensión-vertical.

Ejemplo 5:

```
  Name #EJ5                          Top Dim 3__________ 5__________ 2__________
  ------------------------------------------------------------------------------
  Dimensions                      Occurrences    Starting from     Spacing
  1 . Index vertical              3__            ________          1   Lines
  2 . Index horizontal            5__            ________          1   Columns
  3 . Index (h/v) V               2__            ________          0   Cls/Ls
```

Arreglo con tres dimensiones. Vertical de 3 Occ, horizontal de 5 Occ y 2 Occ por cada Occ (h/v). Un espacios en blanco entre cada Occ vertical y un espacios en blanco entre cada Occ horizontal. El arreglo será recorrido en el orden siguiente, primero dimensión-vertical, luego dimensión-horizontal y por último la dimensión (h/v).

Salida de ejemplos 1, 2 y 3 de arreglos con formato y longitud (A2):

```
+XXXXXXXX                                                          +XXXXXXXX
+XXXXXXXX            ?ACTUALIZACION?DE?DATOS?DE?EMPLEADOS          +XXXXXXXX
                     ?------------?--?-----?--?---------
  EJ1: XX   EJ2: XX XX XX XX XX XX XX XX XX XX     EJ3: XX XX XX XX XX

       XX                                              XX XX XX XX XX

       XX                                              XX XX XX XX XX

       XX                                              XX XX XX XX XX

       XX                                              XX XX XX XX XX

       XX                                              XX XX XX XX XX

       XX                                              XX XX XX XX XX

       XX                                              XX XX XX XX XX

       XX                                              XX XX XX XX XX

       XX                                              XX XX XX XX XX

       XX                                              XX XX XX XX XX


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -     +     Full  <     >     Let
001   --010---+----+----+---030---+----+----+---050---+----+----+---070---+----
```


Autor: Yorman Guadarrama Martínez