

Tipos de Datos

Educating for life

Programación III

Analista de Sistemas de Computación

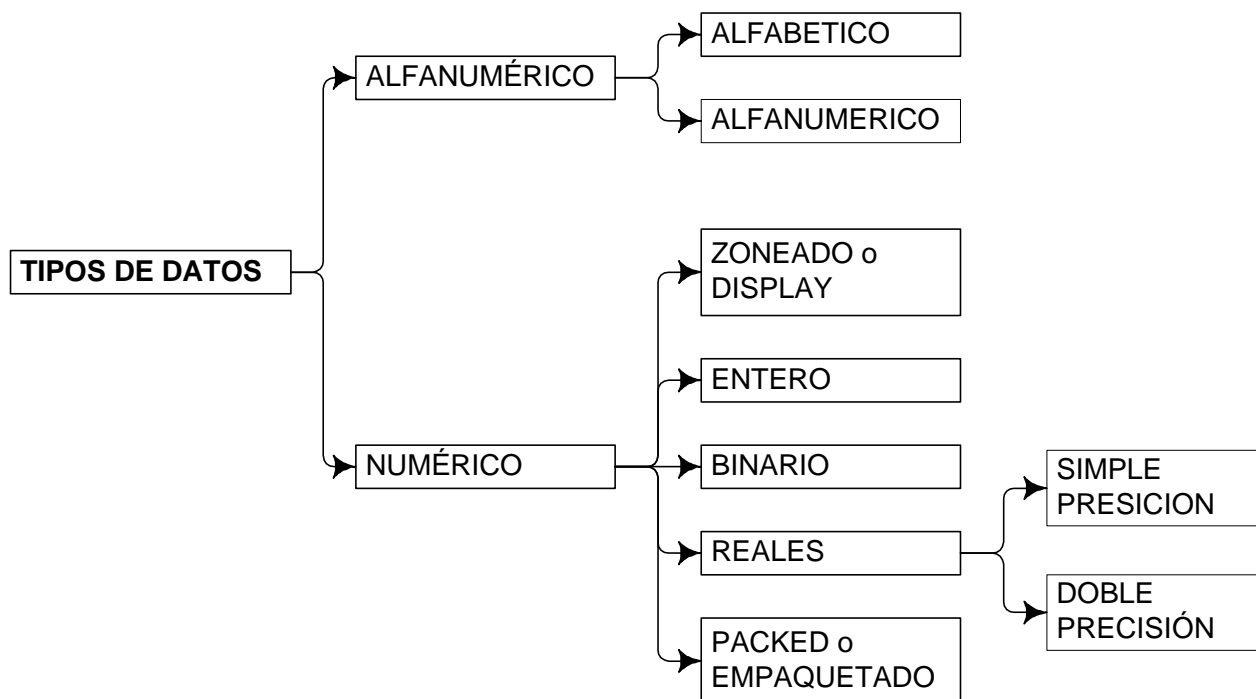
TIPOS DE DATOS

Un tipo de dato se distingue por el conjunto de valores que pueden ser almacenados en el mismo, por las operaciones que se pueden efectuar con ellos y por la forma en que los mismos quedan representados en la memoria. Así, dependiendo de cada lenguaje, podemos hablar de datos de tipo carácter (o String), numéricos, lógicos, etc.

Trabajando sólo con los tipos de datos más comunes, podemos hacer una primera gran división hablando de los datos numéricos y de los alfanuméricos, la gran diferencia entre ellos es que, con los datos numéricos podemos efectuar operaciones matemáticas y con los alfanuméricos no.

Los campos alfanuméricos pueden luego ser divididos en los alfabéticos (que solamente pueden contener las letras del alfabeto) y los alfanuméricos que no sólo pueden contener las letras del alfabeto sino también la mayoría de los caracteres del set de caracteres con los que se este trabajando (EBCDIC, ASCII, Etc.).

También podemos dividir a los campos numéricos, existen diversos formatos de los mismos y, de acuerdo a cada formato será la representación que los mismos tendrán en la memoria o el dispositivo en que se graben.



Veamos la forma en que podríamos dividir los distintos tipos de datos y continuemos luego describiendo cada uno de ellos.

Veamos ahora que contendrá y como será la representación de cada uno de los tipos de datos del diagrama en la memoria de la computadora. Para esto debemos primero saber que la memoria se divide en posiciones fija (los bytes) y que éstas a su vez se dividen en las unidades básicas de información (los bits). A los efectos prácticos los bits se agrupan para formar dos medio bytes (nibbles), cada una de estas mitades cumplen una función particular y se las denomina **ZONA** a la primera mitad (la que tiene los bits con mayor valor relativo) y **DÍGITO** a la segunda mitad (la de valores relativos más bajo). Cada una de estas mitades pueden tomar los valores hexadecimales 0 a F.

Alfabético / Alfanumérico

Estos tipos de datos se diferencian entre sí en que, mientras que los alfabéticos sólo pueden contener letras, los alfanuméricos pueden contener letras, números, signos de puntuación y otros diversos caracteres especiales que pueden ser procesados por la computadora. Cada byte contendrá un carácter, y la estructura binaria de los mismos dependerá del set de caracteres que se esté utilizando.

Para ASCII tomará los valores H²30 a H39 (decimal 48 a 57) para los números positivos, H70 a H79 (112 a 121) para los negativos (que se confunden con las letras “p” a “y”), H41 a H5A (65 a 90) para las letras mayúsculas y H61 a H7A (97 a 122) para las minúsculas. Como podrá observarse para pasar de mayúsculas a minúsculas se le suma H20 (32) y para pasar de números positivos a negativos se le suma H40 (64).

Para EBCDIC tomará los valores HF0 a HF9 (decimal 240 a 249) para los números, HD0 a HD9 (208 a 217) para los números negativos (que se confunden con las letras “J” a “R”), H81 a H89 (129 a 137) para las letras “a” a “i”, H91 a H99 (145 a 153) para las letras “j” a “r” y HA2 a HA9 (162 a 169) para las letras “s” a “z”, HC1 a HC9 (193 a 201) para las letras “A” a “I”, HD1 a HD9 (209 a 217) para las letras “J” a “R” y HE2 a HE9 (226 a 233) para las letras “S” a “Z”. Como podrá observarse para pasar de mayúsculas a minúsculas se le resta H40 (64) y para pasar de números a negativos se le resta H20 (32), los números con signo positivo se representan en este set de caracteres por las combinaciones HC0 a HC9 (192 a 201) es decir H10 (16) menos que los negativos.

Numéricos


Como ya hemos visto hay numerosos tipos de datos que se incluyen dentro de esta categoría. La característica común a todos ellos es la de contener valores con los que podemos efectuar operaciones aritméticas. Para ello todos pueden contener números y, dependiendo de su estructura, los signos (+, -) y los caracteres de puntuación (. y ,). La manera en que se almacenan en la memoria depende del tipo de dato particular que se trate, veamos los principales en forma separada.

² H - Hexadecimal.

Zoneado (ó Display)

Este tipo de datos se caracteriza por tener completos para cada número tanto la parte de zona como la del dígito, en la zona del último número esta el signo, en los anteriores puede tener cualquiera de los dos signos. Veamos entonces como quedarían representados algunos números en ASCII y EBCDIC.

Para representar el número 12345 (positivo o sin signo) en ASCII tendríamos

 Zona

3	1	3	2	3	3	3	4	3	5
---	---	---	---	---	---	---	---	---	---

 Dígito

Para representar el número -12345 (negativo -> 1234u) en ASCII tendríamos

3	1	3	2	3	3	3	4	7	5
---	---	---	---	---	---	---	---	---	---

Para representar el número 12345 en EBCDIC tendríamos

F	1	F	2	F	3	F	4	F	5
---	---	---	---	---	---	---	---	---	---

Para representar el número +12345 (positivo -> 1234E) en ASCII tendríamos

F	1	F	2	F	3	F	4	C	5
---	---	---	---	---	---	---	---	---	---

Para representar el número -12345 (negativo -> 1234N) en ASCII tendríamos

F	1	F	2	F	3	F	4	D	5
---	---	---	---	---	---	---	---	---	---

Binario (Binario Puro ó Binario Entero)

Es como todos sabemos, el sistema de numeración en que está basado el almacenamiento de la computadora, consiste de una serie de caracteres **0** y **1** que representan el valor de la potencia de dos de acuerdo a la posición que ocupen. Dependiendo de los números a almacenar y si tienen signo o no se calcula el tamaño de los binarios en la memoria de la computadora o el dispositivo en que se almacenen, así, en un byte se pueden almacenar números que vayan de -128 a +127 ó de 0 a 255, en dos bytes podemos almacenar desde -32768 a +32767 ó de 0 a 65535. Casi todos los lenguajes usan, para almacenar números binarios 1, 2, 4, u 8 bytes. Siempre que un número binario tenga signo este será el primero de la izquierda siendo el **0** positivo y el **1** negativo y su estará en forma de complemento. Veamos ahora la representación de algunos números en binario.

Para representar el número 12345 (o +12345) tendríamos

ó

3	0	3	9
---	---	---	---

001	000	001	100
1	0	1	1

Para representar el número -12345 tendríamos

ó

C	F	C	7
---	---	---	---

110	111	110	011
0	1	0	1

Reales (Simple y Doble Precisión)

Estos tipos de datos también están representados en binario, la diferencia entre ambos es la cantidad de dígitos significativos que pueden contener. Y su formato difiere de los binarios puros y es particular, están compuestos de dos partes el exponente y el número propiamente dicho, el exponente es la potencia de 10 a la que hay que elevar el número y el número siempre esta representado como fracción, así, el número 12345 queda representado por $0,12345 * 10^5$.

La representación completa del número tiene las siguientes partes:

- 1 Bit que indica el signo del número (0 +, 1 -).
- 7 Bits para indicar el exponente al que hay que elevar el número, se lo toma como un número entero sin signo y se le resta 64 al valor del mismo, así con 7 bits podremos representar los números de 0 a 127, lo que nos permite tener exponentes que van de -64 a +63 aproximadamente.
- Los restantes 3 o 7 Bytes (24 a 56 bits) según sea simple o doble precisión, se utilizan para indicar la fracción es decir el número o, como se lo denomina, mantisa.

De lo dicho, el número 12345 estará representado por el signo + la mantisa **0,12345** y el exponente **5** es decir **+0,12345E5**. Todo esto representado en valores binarios. Veamos entonces como quedarían representados algunos números.

Para representar el número 12345 (o +12345) tendríamos

0	45	1F	9A	6B
---	----	----	----	----



Signo



Exponente



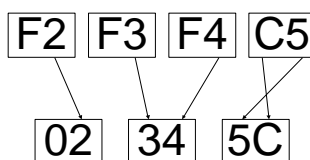
Mantisa

ó

0 1000101	00011111	10011010	01101011
--------------	----------	----------	----------

Empaquetados (ó Packed)

Este tipo de números permite hacer “paquetes” con dos números y poner a estos en un solo byte, para ello trabaja de la siguiente forma, toma el último número (en formato zoneado) e invierte el DÍGITO y la ZONA, luego de cada uno de los números anteriores toma la parte de DÍGITO y, con cada dos de estos forma un byte, por último, si es necesario, agrega ceros a la izquierda para completar la longitud del campo. Veamos como es esto:



OTROS tipos de datos que pueden ser manejados, dependiendo de los lenguajes, son:

DATE: Permite guardar fechas, el formato en que se muestra o recibe la información depende de la forma en que estén seteadas algunas variables del sistema, internamente se guardan como datos numéricos con formato AAAAMMDD y permiten, en general, efectuar operaciones de suma y resta entre ellas o con números naturales.

LÓGICAS: Pueden tomar valores que corresponden a verdadero y falso, casi todos los lenguajes los resuelven convirtiendo a los mismos en valores numéricos correspondiendo el 0 (cero) a verdadero y el 1, o cualquier otro valor, a falso.

PUNTEROS: Este tipo de datos permite almacenar direcciones de memoria a las que apuntan. El lugar de memoria que se apunta es normalmente otro dato aunque también pueden ser apuntados por tipo de variables los nombres de párrafo, es decir el comienzo de rutinas.

MEMO: Son datos de tipo carácter, pero permiten almacenar un número muy grande de caracteres, normalmente se implementan guardando en los mismos un puntero a los datos, los datos en sí están almacenados en otro archivo y se accede a los mismos por medio de instrucciones que se encargan de convertir la dirección y buscar los datos en el archivo correspondiente.

TIPOS DE VARIABLES

Los lenguajes permiten, en general, definir distintos tipos de variables. Los distintos tipos de variables se caracterizan por el ámbito en el que pueden ser utilizadas, es decir el período de vida que poseen, veamos algunos de los distintos tipos de variables.

Locales

Este tipo de variables sólo es visible dentro del procedimiento o función en las que se las utiliza. Si existe una variable del mismo nombre declarada fuera del procedimiento o función esta no se verá afectada por las operaciones que hagamos sobre la variable LOCAL. Si una variable local es usada en un procedimiento recursivo cada paso del procedimiento crea UNA NUEVA instancia de la variable, por lo que al volver la variable no se verá afectada.

De lo antes dicho se ve que las variables LOCALES solo viven (tienen vigencia) dentro de una instancia definida de un procedimiento o función, por lo que cada vez que se ejecute el procedimiento o función será una nueva variable la que se está utilizando.

Por otra parte al terminar de ejecutarse el procedimiento o función que declara una variable LOCAL la misma deja de existir liberando el área de memoria que estaba utilizando.

Normalmente estas variables deben ser declaradas explícitamente y en el comienzo del procedimiento o función que la utiliza.

Globales

Este tipo de variables son visibles en todo momento y por cualquier procedimiento ó función que se este ejecutando, si un procedimiento ó función altera el valor de una variable GLOBAL, el valor de la misma se vera afectado para todos los demás procedimientos ó funciones que la utilicen.

De lo antes dicho se ve que las variables GLOBALES viven desde el momento en que son declaradas y hasta que termine la ejecución del último procedimiento o función del programa en que son utilizadas.

Normalmente este tipo de variables se declara al comienzo del programa en lo que sería la rutina principal del mismo, o en un procedimiento declarado a tal efecto.

VARIABLES Y DATOS EN FOXPRO

En Foxpro se puede declarar una variable en cualquier momento y poner en la misma cualquier tipo de dato, así, una variable que ahora usamos para contener fechas la podríamos usar luego para contener números y más tarde para caracteres, esto no es lógico ni aceptable en un ambiente de programación ordenado, por esto se recomienda declarar explícitamente las variables dentro de un programa FOXPRO y respetar el tipo de datos que las mismas contienen.

Por otro parte es recomendable dar a las variables un nombre que identifique claramente a que tipo pertenece, respetando el largo de 10 caracteres que permite el lenguaje, se puede utilizar el primero para indicar el tipo así, un nombre apropiado para una variable numérica que contendrá el número de cliente podría ser **nNroCli**; Como se ve, el primer carácter nos indica el tipo de variable (n = Numérico) y luego usamos las letras mayúsculas para indicar el inicio de cada palabra o abreviatura que compone el nombre por lo que tenemos dos palabras abreviadas como Nro. (número) y Cli (cliente).

Los sufijos utilizables podrían ser **n**, como ya vimos para numéricos, **c** para los caracteres, **m** para campos memo, **d** para campos de fecha (date), **l** para campos lógicos, también podríamos utilizar **a** (array) para las declaraciones de vectores y matrices que hagamos, en este caso podría haber una segunda letra (alguna de las anteriores) que indique el tipo del array.

Veamos ahora los tipos de variables que nos permite utilizar el FOXPRO.

TIPOS DE VARIABLES EN FOXPRO

FOXPRO clasifica las variables de memoria en cuatro tipos

- ◆ Locales
- ◆ Estáticas
- ◆ Privadas
- ◆ Públicas

Locales

Una variable local necesita ser declarada por un comando LOCAL, este debe ser el primer comando ejecutable contenido en el programa, rutina o procedimiento en que se declare la variable. Estas variables se “crean automáticamente” cada vez que el programa, procedimiento ó función en que se declaren es activado, así para un procedimiento recursivo existirán distintas instancias de la variable. De existir una variable declarada anteriormente con el mismo nombre esta no será afectada por los cambios efectuados a la variable local. Viven y mueren con el procedimiento que las crea.

Privadas

Son variables que se ocultan al programa actual y que estaban definidas por el programa llamador (el que llamó al programa actual).

Públicas

Este tipo de variables no necesitan ser declaradas como tales, si se lo hace se utiliza el comando PUBLIC. Se crean dinámicamente en tiempo de ejecución y siguen existiendo mientras no termine la ejecución del programa en el que fueron declaradas o se utilice algún comando para liberarlas.

A diferencia de las variables privadas, estas son visibles por todos los procedimientos del programa y que este llama, normalmente se declaran en el cuerpo principal del programa.

TIPOS DE DATOS EN FOXPRO

Como ya dijimos, en FOXPRO no es necesario definir el tipo de datos que contendrá una variable, es más, podríamos variar el tipo de una variable entre un punto y otro del programa, ahora bien, esto no es aconsejable y llevaría a la confusión por lo que no debemos hacerlo y deberíamos utilizar una nomenclatura tal como la que hemos mencionado para que el programa sea entendible y fácil de mantener.

Por otra parte este lenguaje no diferencia, al menos para el usuario o programador, los distintos tipos de datos numéricos en la forma en que lo hemos mencionado, así tendremos solo datos numéricos sin importar el tratamiento que le vayamos a dar a los mismos.

Los tipos de datos manejados por FOXPRO son:

- **CHARACTER:** Estos datos pueden contener todos los caracteres ASCII que no posean funciones especiales (en general los caracteres cuyos valores ASCII son menores a 32 funcionan como caracteres de control aunque algunos de ellos pueden de todas formas ser usados en textos), la longitud de estos datos puede alcanzar los 254 caracteres.
- **MEMO:** Estos datos son en su formato similar a los de tipo carácter, no así su tratamiento ni su longitud. En lo que respecta a la longitud pueden contener una cantidad VARIABLE de caracteres y puede exceder ampliamente los 64K, en lo que respecta a su tratamiento poseen instrucciones particulares del lenguaje que permiten su edición, almacenamiento, extracción y modificación. Por otra parte, en el archivo original solo se almacena un puntero (10 posiciones) que indican donde están los datos, estos se almacenan en un archivo aparte del que son recuperados utilizando el puntero.
- **DATE:** Estos datos se utilizan para almacenar datos que contendrán fechas, el formato en que se visualizarán depende del seteo que tengan varias variables del

lenguaje (set date, set century) pero indistintamente de las mismas se almacenan en formato AAAAMMDD, es posible efectuar operaciones de suma y restas con este tipo de variables entre sí o con números naturales.

- **NUMERICOS:** Este tipo contiene datos que pueden ser usados en operaciones matemáticas, su longitud puede llegar a los 32 caracteres (30 dígitos + signo + punto decimal), y puede tomar valores que van de 10^{-308} a 10^{308} ; sin embargo solo se puede asegurar una precisión de 16 dígitos, esto significa que, números que necesitan más de 32 posiciones para ser mostrados se displayan como asteriscos y, los dígitos que excedan los 16 dígitos representativos son mostrados como ceros; el formato en que se lo visualiza puede variar de acuerdo con la máscara de edición que se utilice.
- **LÓGICO:** Este tipo de datos asume valores booleanos (Verdadero y Falso), puede contener, de acuerdo a la máscara que se utilice, los caracteres Y, y, N, n, T, t, F, f, algunas versiones que se utilizan en castellano permiten S, s en lugar de Y, y. (Y, y, T, t, S, s representan los valores verdaderos; N, n, F, f representan los valores falsos).
- **ARRAY:** En FOXPRO las estructuras Array son tratadas como un tipo separado de datos y existen operaciones sobre el tipo array que son inválidos para otros tipos de datos. Este tipo será tratado en el momento en que se los utilice.

TABLA DE TIPOS DE DATOS DE VISUAL FOXPRO

Tipo de datos	Descripción	Tamaño	Intervalo
Carácter	Cualquier texto	1 byte por carácter hasta 254	Cualquier carácter
Currency	Cantidades monetarias	8 bytes	– 922337203685477.5808 a 922337203685477.5807
Date	Datos cronológicos formados por mes, año y día	8 bytes	01/01/100 a 12/31/9999
DateTime	Datos cronológicos formados por mes, año, día y hora	8 bytes	01/01/100 a 12/31/9999, más 00:00:00 a.m. a 11:59:59 p.m.
Logical	Valor booleano verdadero o falso	1 byte	Verdadero (.T.) o Falso (.F.)
Numeric	Enteros o fracciones	8 bytes en la memoria; 1 a 20 bytes en una tabla	– .9999999999E+19 a .9999999999E+20

TABLA DE TIPOS DE CAMPOS DE VISUAL FOXPRO

Tipo de datos	Descripción	Tamaño	Intervalo
Double	Número de coma flotante de precisión doble	8 bytes	+/- 4.94065645841247E-324 a +/-8.9884656743115E307
Float	Igual que Numeric	8 bytes en la memoria; 1 a 20 bytes en una tabla	– .9999999999E+19 a .9999999999E+20
General	Referencia a un objeto OLE	4 bytes en una tabla	Limitado por la memoria disponible
Integer	Valores enteros	4 bytes	–2147483647 a 2147483646
Memo	Referencia a un bloque de datos	4 bytes en una tabla	Limitado por la memoria disponible
Character (Binario)	Datos de tipo carácter que quiere mantener sin modificación en páginas de códigos	1 byte por carácter hasta 254	Cualquier carácter
Memo (Binario)	Datos de campo memo que quiere mantener sin modificación en páginas de códigos	4 bytes en una tabla	Limitado por la memoria disponible

VARIABLES Y DATOS EN COBOL

En COBOL no hay una división exacta entre los tipos de variables, es así que no se habla de variables locales, globales, etc.

A pesar de esto la “**SECTION**” en que se declare una variable está asociada a la vida y uso que tendrá la misma, por esto en lugar de referirnos a los tipos de variables mencionaremos las secciones en que se declaran las variables en cobol y su uso.

- **FILE SECTION:** Las variables declaradas en esta sección son utilizadas para la transferencia de datos desde y hacia los archivos y por lo tanto tendrán vida dentro de los mismos y, una vez en memoria, hasta que sean reemplazadas por las variables de otro registro del archivo.
- **WORKING-STORAGE SECTION:** En esta sección se declaran todas las variables de trabajo del programa, su vida y ámbito corresponden a todo el programa.
- **LINKAGE SECTION:** En esta región se definen las variables que corresponden a los parámetros recibidos desde un programa llamador, viven y pueden ser usadas a lo largo de todo el programa.
- **SCREEN SECTION:** En esta sección se definen las variables que se utilizarán como entrada y salida a pantallas, su formato así como uso son muy distintos del de los demás tipos de variables. No todas las versiones de cobol poseen esta sección.
- **LOCAL-STORAGE SECTION:** Esta sección existe solo en unas pocas versiones de cobol, en estas versiones el cobol permite la recursividad y, estas variables son las que permiten ser usadas en las distintas instancias de un programa recursivo, es decir existe una instancia de la variable por cada llamada del procedimiento.
- **COMUNICATION SECTION:** Esta sección describe las variables que se usarán para comunicarse (por lo general vía módem) con otros equipos, estas serán transmitidos o recibidos desde los mismos. Son pocas las versiones de cobol que poseen esta sección.
- **REPORT SECTION:** Define las variables que serán utilizadas por un proceso especial de generación de reportes dentro de los programas cobol.

TIPOS DE DATOS EN COBOL

En COBOL es necesario definir cada entrada de datos que vayamos a utilizar, indicando no solo su nombre sino también el tipo de datos que contendrá, el tamaño del mismo y su formato. Necesitamos además conocer que tipos de datos pueden operar entre sí, que sucede cuando operamos con cada uno de estos tipos de datos (lo que se verá luego en unas tablas) y como redefinir, si es necesario, una variable para que puedan efectuarse las operaciones mencionadas en forma correcta.

En COBOL, cada ítem de dato (o descripción de entrada de datos) tiene una jerarquía determinada, cada uno de estos ítems le da nombre, reserva memoria para el almacenamiento y describe al dato que almacenará.

La jerarquía a cada ítem se la da el **número de nivel**, los números de nivel van del 01 (el de mayor nivel) a 48 (el nivel más bajo), en este orden jerárquico un nivel es “padre” de los niveles inferiores a él (número mayor) que contenga y contiene en sí mismo a todos ellos. De este modo el nivel 01 es padre y contiene a todos los niveles inferiores.

Existen dos tipos de ítems de datos, aquellos que definen datos en sí mismos y aquellos que, sin definir datos en sí mismos agrupan ítems de menor nivel, a los primeros se los denomina ítems de datos ELEMENTALES, los segundos son conocidos como ítems DE GRUPO.

Además de los niveles antes mencionados (01 a 48) existen otros niveles de datos en Cobol 77, 88 y otros que tienen una función particular, por el momento solo veremos la función de los dos mencionados explícitamente.

Nivel 77: define un ítem de dato elemental, no puede ser padre de otros niveles, y por esto es equivalente, con algunas ventajas que luego se mencionarán, y reemplaza a un nivel 01 que no se define como ítem de grupo.

Nivel 88: este nivel no define un dato en sí mismo, lo que hace es definir valores lógicos para las variables en que se encuentra definido que permiten el testeo por verdadero o falso de los valores que define contra el contenido de la variable ítem en que se encuentra.

Sería difícil, y por otro lado innecesario en este momento, definir completamente una entrada de datos, es por esto que definiremos el formato que más usaremos dejando para luego, en la medida que sea necesario, los otros formatos. Veamos entonces el formato más general que, por otra parte, abarca a casi todas las definiciones que utilizaremos, para definir el formato usaremos **NEGRITAS** para las partes obligatorias, *itálicas* para los campos que debe definir el programador y [] para definir las partes opcionales de la cláusula. Con esto el formato es:

NIVEL *NOMBRE1* [**REDEFINES** *NOMBRE2*] [**OCCURS** *VALOR1* [**TO** *VALOR2*] [**TIMES**]
[**DEPENDING ON** *NOMBRE3*]] **PIC**[**TURE**] *MASCARA* [[**USAGE IS**] *TIPO*]

Donde:

NIVEL - Es el número que indica el nivel del ítem según ya hemos visto.

NOMBRE1 - Es el nombre del campo (ítem de datos) que queremos definir, Se puede utilizar la cláusula FILLER como nombre, con esto indicamos que el campo no posee un nombre definido.

REDEFINES - Usaremos esta cláusula si estamos redefiniendo (dando otro formato) a un campo que ya hemos definido; el campo que vayamos a definir debe tener el mismo nivel que aquel al que estamos redefiniendo.

NOMBRE2 - Es, si usamos REDEFINES, el nombre del campo que estamos definiendo y que modifica el formato del anterior.

OCCURS - Usamos esta cláusula para definir un arreglo, por medio de esto indicamos las ocurrencias (cantidad de elementos) que tendrá el arreglo.

NUMERO1 - Este número indica, la cantidad de elementos que tendrá el arreglo, si tendrá una cantidad fija de ocurrencias, ó, la menor cantidad de elementos que tendrá el mismo si la cantidad de ocurrencias es variable.

TO *NUMERO2* - Esta cláusula, opcional en la definición de un campo, indica si aparece que la cantidad de ocurrencias del arreglo es variable y, que ésta será la cantidad máxima de elementos que puede contener el arreglo.

TIMES - Esta cláusula es totalmente opcional y no agrega información a la definición del campo que se está efectuando, se la usa para una lectura más fluída de la definición.

DEPENDING ON *NOMBRE3* - Esta cláusula define, para un arreglo de longitud variable, en que campo tendremos la cantidad real de ocurrencias del arreglo.

PIC o **PICTURE** - Nos indica que el valor que lo sigue es una "máscara" que nos indicará el formato y longitud del campo que estamos definiendo.

MASCARA - Define, entre otros atributos, la longitud del campo y el tipo de datos que podrá ser almacenado en la variable que estamos definiendo. Veremos más adelante el formato y uso de esta máscara.

USAGE IS - Esta cláusula, opcional, nos indica que usaremos un "tipo" de datos particular para el campo que estamos definiendo, esto implicará un formato particular en la memoria de la computadora o en el dispositivo de almacenamiento, si no se indica el tipo de datos es el normal para el campo que estemos definiendo. Puede no ponerse explícitamente la cláusula usage is pero si indicarse un tipo, ambas formas son equivalentes.

TIPO

- Define, tal como hemos dicho, una forma particular de almacenar los datos en la variable que estamos definiendo. Luego veremos los distintos tipos de datos que se pueden definir, estos corresponden, en Cobol, a los ya vistos anteriormente en forma general.

CARACTERES PERMITIDOS EN LA DEFINICIÓN DE UNA “MASCARA”

Veremos ahora los caracteres que podemos utilizar en la definición de la máscara de la cláusula PICTURE, indicando en cada caso su significado y en que campos pueden ser utilizados, luego daremos algunos ejemplos de su uso.

Tipo de Carácter	Símbolo Usado	Significado o Uso
Carácter para Definición	9	Campo Numérico
	A	Campo Alfabético
	X	Campo Alfanumérico
Caracteres Especiales	V	Punto Decimal supuesto
	P	Escalamiento Decimal
	S	Inclusión del Signo
Caracteres de Edición	\$	Inserta Signo Pesos
	Z	Cambia Ceros por Blancos
	*	Cambia Ceros por *
	.	Punto decimal
	,	Coma decimal
	+	Signo (positivos y negativos)
	-	Signo (solo negativos)
	DB	Débito (negativos)
	CR	Crédito (negativos)
	B	Inserta blancos
	0	Inserta Ceros
	/	Pone el carácter /

Veamos algunos ejemplos del uso de estos caracteres, en los mismos la posición de la coma (punto decimal) del almacenamiento se indicara con ^.

Carácter V

Este carácter indica la posición de un punto decimal supuesto, es decir no escrito, lo que implica su consideración en cualquier operación de cálculo.

Ej.:

PICTURE	VALOR	ALMACENAMIENTO
PIC 9(2)V9	38,50	38^5
PIC 9(4)V99	459,61	459^61
PIC 9(3)V9(3)	459,61	459^610

Carácter P

Este carácter, usado junto con el anterior, indica la posición de un punto decimal, cuando se entiende que los datos son en miles, múltiplos o submúltiplos de 1000.

Ej:

PICTURE	VALOR	ALMACENAMIENTO
PIC 99PPV9	12	12000^
PIC VP(3)9(4)	1023	^0001023

Carácter S

Este carácter se utiliza para definir una variable con signo, al procesar la información se vale de otro carácter, si el signo se pone por separado, o de un método de chequeo del signo de la información (bit, zona o posición del signo) para tratar la información como positiva o negativa según corresponda al caso.

Ej:

PICTURE	VALOR	ALMACENAMIENTO
PIC S9(4)V99	-1251	-1251^00
PIC S9(4)	+1251	+1251^

El signo estaría en un dígito aparte, como zona del último dígito o como bit de signo.

Carácter X

Este carácter se utiliza para definir variables alfanuméricas. Se completan con espacios los lugares que sobren a la derecha de la variable o, si por el contrario es más corta que la información a contener, se truncan los caracteres derechos de la información a almacenar.

Ej:

PICTURE	VALOR	ALMACENAMIENTO
PIC x(15)	"Prueba"	Pruebabbbb
PIC x(10)	"CARACTERES"	CARACTERES
PIC X(5)	"AXZ\$123456"	AXZ\$1

Carácter A

Este carácter se utiliza para alfabéticas, solo permite almacenar las letras el alfabeto, por lo demás su funcionamiento y uso es igual al indicado para el carácter "X".

Caracteres \$, + y -

Estos caracteres se utilizan en las cláusulas de edición para indicar la presencia de los símbolos \$, + y - respectivamente, su uso se observará mejor a través de los ejemplos que se dan a continuación.

Ej:

PICTURE	VALOR	ALMACENAMIENTO
PIC +999.9	35,2	+035.2
PIC 999.9+	35,2	035.2+
PIC 999.9-	-35,2	032.2-
PIC 999.9-	35,2	035.2
PIC 999.9-	-35,2	035.2-
PIC ++9.9	001,3	+1.3
PIC ++9.9	-1,3	-1.3
PIC ++9.9	-0,5	-0.5
PIC ++9.9	20	+20.0
PIC +++++.+	0,1	+1
PIC +++++.+	0	+0
PIC ----	0	
PIC --99.9	-10,25	-10.25
PIC --99.9	10,25	10.25
PIC 999.9-	10,2	010,2
PIC \$99.99-	2,3356	\$ 2,33
PIC \$\$\$99-	2,3356	\$2.33
PIC \$\$\$99-	-2,3356	\$2.33-
PIC \$\$\$99+	2,3356	\$2.33+
PIC \$\$\$99+	-2,3356	\$2.33-

Nótese que en los campos de edición se usan el punto y la coma en lugar del carácter V de los casos anteriores.

La longitud del campo es siempre la indicada en la máscara completándose con blancos a izquierda.

Caracteres DB y CR

Estos caracteres se utilizan para aplicaciones contables y representan débito y crédito respectivamente.

Ej:

PICTURE	VALOR	ALMACENAMIENTO
PIC \$999.99DB	135,26	\$135.26
PIC \$999.99CR	-135,26	\$135.26CR
PIC \$,\$99.99DB	-10,50	\$10,50DB

Carácter B

Este carácter se utiliza para agregar espacios en blanco.

Ej.:

PICTURE	VALOR	ALMACENAMIENTO
PIC ABABAAAA	"NYAPEL"	N Y APEL
PIC 99B99B99	123456	12 34 56

Carácter 0

Este carácter se utiliza para agregar ceros en la posición indicada.

Ej:

PICTURE	VALOR	ALMACENAMIENTO
PIC 9(4)0000	365	3650000

Carácter /

Este carácter se utiliza principalmente con fechas para darle el formato adecuado a las mismas.

Ej:

PICTURE	VALOR	ALMACENAMIENTO
PIC 99/99/99	230197	23/01/97

Carácter *

Este carácter se utiliza en formularios para editar las cifras a fin de evitar su adulteración, se usa como carácter de relleno en cheques y otros documentos.

Ej:

PICTURE	VALOR	ALMACENAMIENTO
PIC \$***.99	110,4	\$110.40
PIC \$***.99	11,1	\$*11.10
PIC \$***.99	1	\$**1.00

TIPOS POSIBLES DE UTILIZAR CON LA CLAUSULA USAGE

Veremos aquí algunos de los formatos permitidos en el tipo de la cláusula USAGE, los mismos pueden sufrir alguna pequeña variación en su significado dependiendo de que cobol estemos utilizando, otros no en todas las versiones del cobol están definidos, lo que definiremos es el significado más común de los mismos en aquellos coboles que los posean.

COMPUTATIONAL o **COMP**: Define campos que tiene el formato interno equivalente a binarios enteros.

COMPUTATIONAL-3 o **COMP-3**: Define campos cuya estructura interna es la que hemos visto como decimal empaquetado (o packed) (también se lo denomina packed-decimal en lugar de comp-3).

COMPUTATIONAL-5 o **COMP-5**: Define campos binarios enteros pero que serán almacenados invirtiendo el orden de los bytes dentro del mismo, esta es la forma en que trabaja normalmente la computadora y por lo tanto los hace más rápidos en el momento de trabajo.

COMPUTATIONAL-1 o **COMP-1**: No todos los coboles lo poseen, en los que lo permiten define campos que son equivalentes a los que hemos visto como simple precisión.

COMPUTATIONAL-2 o **COMP-2**: Al igual que el anterior no todos los coboles lo poseen y define campos de los que hemos visto como doble precisión.

COMPUTATIONAL-X o **COMP-X**: Cumple función similar a comp-1 y comp-2, en realidad generalmente los reemplaza, la diferencia es que toma simple o doble precisión dependiendo de los valores a almacenar en él (según se defina en el pic).

DISPLAY: Es el default, es decir el que asume si no se le indica nada, y equivale a los campos que hemos visto como zoneados.

Existen otros tipos (INDEX, POINTER, DISPLAY-1, PROCEDURE_POINTER) que no discutiremos en este momento, los mismos quedan para ser tratados a posteriori o investigado por Uds. si tienen necesidad de usarlos.

A continuación detallaremos dos tablas que nos permitirán ver los movimientos posibles entre los distintos campos así como las comparaciones que podemos hacer con los mismos.

MOVIMIENTOS ENTRE CAMPOS

Campo fuente	Campo receptor						
	GR.	AL.	AN.	DEC.	BIN.	PF.	RP.
Ítem de grupo (GR.)	SI	SI	SI	NO	NO	NO	NO
Alfabético (AL)	SI	SI	SI	NO	NO	NO	NO
Alfanumérico (AN)	SI	SI	SI	NO	NO	NO	NO
Decimales internos y externos (DEC) (display, comp-3 ...)	SI	NO	SI ²	SI	SI	SI	SI
Binarios (BIN) (comp, comp-5, comp-x, ...)	SI	NO	SI ¹	SI	SI	SI	SI
Punto flotante (comp-1, comp-2, ...)	SI	NO	NO	SI	SI	SI	SI
Informe (RP)	SI	NO	SI	NO	NO	NO	NO
Constante figurativa cero (ZERO o ZEROES)	SI	NO	SI	SI	SI	SI	SI
Constante figurativa blancos (SPACE o SPACES)	SI	SI	SI	NO	NO	NO	NO
Otras constantes figurativas ALL "Carácter" HIGH-VALUE LOW-VALUE QUOTES	SI	NO	SI	NO	NO	NO	NO

Campo fuente	Campo receptor							
	GR.	AL.	AN.	DEC.	BIN.	PF.	RP.	CF.
Ítem de grupo (GR.)	NN	NN	NN	NN	NN	NN	NN	NN
Alfabético (AL)	NN	NN	NN					NN ³
Alfanumérico (AN)	NN	NN	NN	NN ⁷			NN	NN
Decimales internos y externos (DEC) (display, comp-3 ...)	NN			NU	NU	NU		NU ⁵
Binarios (BIN) (comp, comp-5, comp-x, ...)	NN			NU	NU	NU		NU ⁴
Punto flotante (comp-1, comp-2, ...)	NN			NU	NU	NU		NU ⁴
Informe (RP)	NN		NN				NN	NN ⁶
Otras constantes figurativas ALL "Carácter" HIGH-VALUE LOW-VALUE QUOTES	NN	NN ³	NN	NU ⁵	NU ⁴	NU ⁴	NN ⁶	

NN - Comparaciones entre ítems NO NUMÉRICOS.

NU - Comparaciones entre ítems NUMÉRICOS.

² Sólo para enteros

¹ Para enteros solamente

³ Permitido con las constantes figurativas SPACE, ALL "carácter" donde carácter debe ser alfabético.

⁷ El campo decimal debe estar formado por enteros.

⁵ Permitido solamente si la constante figurativa es ZERO o ALL "carácter" donde carácter es numérico.

⁴ Permitido sólo si la constante figurativa es ZERO.

⁶ NO permitido con la constante figurativa QUOTE.

INDICE

TIPOS DE DATOS.....	1
Alfabético / Alfanumérico.....	2
Numéricos	2
Zoneado (ó Display).....	3
Binario (Binario Puro ó Binario Entero).....	3
Reales (Simple y Doble Precisión).....	4
Empaquetados (ó Packed).....	5
TIPOS DE VARIABLES	6
Locales	6
Globales	6
VARIABLES Y DATOS EN FOXPRO	7
TIPOS DE VARIABLES EN FOXPRO	7
Locales.....	7
Privadas.....	8
Públicas.....	8
TIPOS DE DATOS EN FOXPRO.....	8
TABLA DE TIPOS DE DATOS DE VISUAL FOXPRO	10
TABLA DE TIPOS DE CAMPOS DE VISUAL FOXPRO	11
VARIABLES Y DATOS EN COBOL.....	12
TIPOS DE DATOS EN COBOL.....	13
CARACTERES PERMITIDOS EN LA DEFINICIÓN DE UNA “MASCARA”	16
Carácter V.....	16
Carácter P.....	17
Carácter S.....	17
Carácter X.....	17
Carácter A.....	17
Caracteres \$, + y -.....	18
Caracteres DB y CR.....	18
Carácter B.....	19
Carácter 0.....	19
Carácter /	19
Carácter *.....	19
TIPOS POSIBLES DE UTILIZAR CON LA CLAUSULA USAGE.....	20
MOVIMIENTOS ENTRE CAMPOS	21
INDICE.....	22