

Aprende ABAP/4 con ejercicios prácticos

Marlon Falcón Hernández
www.marlonfalcon.cl

DERECHOS DEL AUTOR

Aprende ABAP/4 con ejercicios prácticos.

Copyright © 2015 by **Marlon Falcón Hernández**

Todos los derechos reservados. Ninguna parte de este trabajo puede reproducirse o puede transmitirse en cualquier formulario o por cualquier medio, electrónico o mecánico, incluyendo fotocopiado, grabado o por cualquier almacenamiento de información, sin el permiso escrito anterior del dueño de los derechos de propiedad literaria y el publicador.

ISBN- PENDIENTE A PUBLICACION

Para más información por favor contacte con el autor de la obra:
Marlon Falcón Hernández mediante los siguientes correos:

contacto@marlonfalcon.cl
falconsoft.3d@gmail.com

Diseño de Portada:
Marlon Falcón Hernández
Personaje de portada: diseñado por Freepik

Dedicatoria

A mi hijo Marlon con todo el Amor del mundo.

Índice

INTRODUCCION.....	11
CAPITULO 1 INTRODUCCIÓN A SAP ERP.....	12
¿QUÉ ES UN ERP?	12
¿CUÁLES SON LOS PRINCIPALES ERP?	12
¿QUÉ ES SAP?	13
¿QUÉ ES ABAP/4?	13
¿QUÉ ES TIPOS DE PROGRAMAS SE PUEDEN HACER EN ABAP?	14
PRIMEROS PASOS PARA CONOCER LA INTERFACE DE SAP ECC.....	14
CONOCIENDO LA INTERFACE DE SAP	15
CAPITULO 2 PROGRAMACIÓN BÁSICA EN ABAP.....	16
EJERCICIO N°1 - HOLA MUNDO EN ABAP/4. (SE38)	16
EJERCICIO N°2 - COMENTARIOS.....	18
EJERCICIO N°3 - IMPRIMIR VARIAS LÍNEAS CON WRITE.	19
EJERCICIO N°4 - TIPOS DE DATOS EN ABAP.....	19
EJERCICIO N°5 - ASIGNACIÓN DE VALOR A UNA VARIABLE	20
EJERCICIO N°6 - CONSTANTES.....	20
EJERCICIO N°7 - VARIABLES DEL SISTEMA. IMPRIMIR LA FECHA.	21
EJERCICIO N°8 - VARIABLES DEL SISTEMA II.	21
EJERCICIO N°9 - VARIABLE SY-SUBRC.....	21
EJERCICIO N°10 - OPERACIONES CON CARACTERES.	22
EJERCICIO N°11 - EJEMPLO DE USO DE SY-SUBRC	24
EJERCICIO N°12 - EJEMPLO DE USO DE LÍNEAS TIPO TABLA.	24
EJERCICIO N°13 - EJEMPLO REDONDEO VARIOS.	25
EJERCICIO N°14 - OPERACIONES MATEMÁTICA.	25
EJERCICIO N°15 - ABAP DEBUGER.....	26
CAPITULO 3 OPERADORES DE CONDICIÓN.....	28
EJERCICIO N°16 - SENTENCIA CHECK.....	28
EJERCICIO N°17 - SENTENCIA IF ..ELSE	28
EJERCICIO N°18- SENTENCIA CASE	29
EJERCICIO N°19- USANDO EL OPERADOR <> DIFERENTE.	30
EJERCICIO N°20- USANDO EL BETWEEN.....	30
CAPITULO 3 BUCLES.....	31

EJERCICIO N°21 - BUCLE DO	31
EJERCICIO N°22 - BUCLE DO CON EXIT	31
EJERCICIO N°23 - BUCLE WHILE.....	31
CAPITULO 4 PARÁMETROS Y SUBROUTINAS.	33
EJERCICIO N°24 - PARÁMETROS DE ENTRADAS.....	33
EJERCICIO N°25 - SUBROUTINAS INTERNAS.	33
EJERCICIO N°26 - SUBROUTINAS INTERNAS CON PARÁMETROS.	33
EJERCICIO N°27 - SUBROUTINAS EXTERNA CON PARÁMETROS.....	34
EJERCICIO N°28 - PLANTILLA DE ABAP/4.....	34
CAPITULO 5 TABLAS INTERNAS.....	36
EJERCICIO N°29 - TABLAS INTERNAS.....	36
EJERCICIO N°30 - TABLAS INTERNAS ESTRUCTURA GENERAL.....	36
EJERCICIO N°31 - TABLAS INTERNAS CON CABECERA.....	37
EJERCICIO N°32 - TABLAS INTERNAS CONSULTA DE USUARIOS.....	38
EJERCICIO N°33 - TABLA INTERNA SIMPLE.	39
EJERCICIO N°34 - TABLAS INTERNAS CON ÁREAS DE TRABAJO.....	39
EJERCICIO N°35 - TABLAS INTERNAS CON FIELD-SYMBOLS	40
CAPITULO 5 DICCIONARIO DE DATOS	42
EJERCICIO N°36 - ENTRANDO AL DD (SE11)	42
EJERCICIO N°37 - TABLA DE UNA TRANSACCIÓN.	44
EJERCICIO N°38 - CONSULTA PARA UNA TABLA DE DD.....	46
EJERCICIO N°39 - REPORTE AVL CON UNA TABLA DEL DD.	46
EJERCICIO N°40 - FULL REPORTE AVL DE UNA TABLA.	47
EJERCICIO N°41 - CREAR UN DOMINIO.	49
EJERCICIO N°42 - CREAR UN ELEMENTO DE DATO.....	51
EJERCICIO N°43 - CREAR TABLA TRANSPARENTE (SE11).	53
EJERCICIO N°44 - LLENAR UNA TABLA DE DATOS (SE16).....	55
CAPITULO 6 OPEN SQL - CONSULTAS ABAP	56
INTRODUCCIÓN A OPEN SQL.	56
EJERCICIO N°45 - CONSULTA BÁSICA DIRECTA.	56
EJERCICIO N°46 - CONSULTA BÁSICA CON TI.....	57
EJERCICIO N°47 - CONSULTA PRIMERAS 100 FILAS.....	57
EJERCICIO N°48 - CONSULTA CONDICIÓN.....	57

EJERCICIO N°49 - MOSTRAR PRIMERA FILA EN UNA TABLA	58
EJERCICIO N°50 - SELECCIONAMOS TODOS	58
EJERCICIO N°51 - SELECCIONAMOS UN SOLO REGISTRO	58
EJERCICIO N°52 - MÁXIMO, MÍNIMO, CANTIDAD.....	59
EJERCICIO N°53 - SUMA Y PROMEDIO.....	59
EJERCICIO N°54 - BUSCANDO CADENAS QUE CONTENGAN OTRAS. 60	
EJERCICIO N°55 - BUSCANDO CADENAS CON LISTAS. IN.....	60
EJERCICIO N°56 - SELECCIONANDO UN RANGO BETWEEN.....	60
EJERCICIO N°57- MOSTRAR TABLA ORDENADA.	61
EJERCICIO N°58 - MOSTRAR TODOS LOS ICONOS EN SAP	61
EJERCICIO N°59- CONSULTA A DOS TABLAS EN SAP.	61
CAPITULO 7 AMPLIACIONES Y NOTAS.....	63
EJERCICIO N°60 - HOLA MUNDO EN ABAP/4.....	63
CAPITULO 8 PROGRAMACIÓN ORIENTADO A OBJETOS.64	
¿QUÉ ES LA PROGRAMACIÓN ORIENTADO A OBJETOS ?	64
EJERCICIO N°61 - HOLA MUNDO DE POO.	64
EJERCICIO N°62 - POO CON MÉTODOS Y PARÁMETROS.....	65
EJERCICIO N°63 - POO HERENCIA DE CLASES.	66
CAPITULO 8 PROGRAMACIÓN DE DIÁLOGOS	68
EJERCICIO N°64- MENSAJES EN ABAP.....	68
EJERCICIO N°65 - CREANDO UNA TRANSACCIÓN EN SAP (SE93) ...	70
EJERCICIO N°66 - COLORES EN UN REPORTE Z.....	73
EJERCICIO N°67 - VARIOS COLORES EN UN REPORTE Z.....	73
CAPÍTULO 9 DYNPRO	74
INTRODUCCIÓN A UNA DYNPRO.....	74
EJERCICIO N°68- CREACIÓN DE UNA DYNPRO (SE80).....	74
CAPITULO 10 BATCH INPUTS.....	77
INTRODUCCIÓN A UNA BATCH INPUTS SE35	77
CAPITULO 11 FORMULARIOS.	78
TIPOS DE FORMULARIOS EN SAP.....	78
EJERCICIO N°69 - MOSTRAR UN FORMULARIO SAPSCRIPT (SE71). 78	
EJERCICIO N°70 - COPIAR UN FORMULARIO SAPSCRIPT.	80
EJERCICIO N°71 - CREAR UN FORMULARIO CON SAPSCRIPT.	81

EJERCICIO N°59 - EJECUTAR UN FORM SAPSCRIPT.....	85
CAPITULO 12 OBJECT NAVIGATOR	86
INTRODUCCIÓN AL OBJECT NAVIGATOR SE80	86
EJERCICIO N°73 - CREAR GRUPO DE FUNCIONES.	86
CAPITULO 13 IDOCS (INTERMEDIATE DOCUMENT)	87
INTRODUCCIÓN A IDOCS	87
EJERCICIO N°74 - HOLA MUNDO EN ABAP/4.....	87
AGRADECIMIENTOS.....	88
TRANSACCIONES MÁS UTILIZADAS EN SAP.....	89
VARIABLES DEL SISTEMA	91
COMANDO DE LA BARRA	96
TECNOLOGÍAS DE INTERFACES UTILIZADAS EN SAP..	96
WORKBENCH ABAP.	96

INTRODUCCION

Este libro está diseñado de tal forma que con ejercicios prácticos vas conociendo cada parte de la programación ABAP/4, no empieza con teorías de programación sino que con pequeños pasos de bebe va enseñándote todo lo que necesitas para iniciarte en el mundo de la programación. Es importante que al finalizar cada clase la repitas y hagas los ejercicios propuestos en el libro. El nivel aumentará a medidas que vallas avanzado en los ejercicios. Trata siempre de llevarlo contigo porque muchos procedimientos si no lo practicas se te olvidaran y aquí siempre lo podrás refrescar.

Y lo más importante practica mucho, pon toda tu voluntad para que puedas entrar sin problemas al mundo laboral de ABAP/4 para SAP.

CAPITULO 1 Introducción a SAP ERP.

¿Qué es un ERP?

Un **ERP** es un sistema (software informático) de planificación de recursos empresariales, su nombre viene de las siglas en Ingles "**Enterprise Resource Planning**". Los **ERP** gestionan e integran los procesos de las empresas permitiendo tener información actualizada y segura para toma de decisiones.

Un **ERP** por lo general administra compras, ventas, producción, contabilidad y logística.

Los objetivos principales de cualquier ERP son:

- Mantener información actualizada de la empresa con acceso seguro para los usuarios.
- Disminuir los tiempo de gestión empresarial optimizando todos los pasos.

¿Cuáles son los principales ERP?

En el mercado existen varios ERP entre los más conocidos se encuentran:

SAP ERP / SAP ECC / SAP R3 - Es uno de los lideres en la lista de los ERP, está enfocado a grandes empresas, aunque puede ser implementado en empresas de cualquier tamaño, tiene una gran escalabilidad ya que trae ABAP/4 como lenguaje de programación que le permite poder programar dentro del sistema cualquier necesidad de la empresa.

SAP BUSINESS ONE - Es uno de los ERP más usados en las pequeñas y medianas empresas, fue creado en Israel y posteriormente fue adquirido por SAP.

Microsoft Dynamics - Es uno de los ERP más usados en las pequeñas empresas propiedad de Microsoft entre sus ventajas esta la gran compatibilidad con varios dispositivos. ofrece cuatro productos ERP: Microsoft Dynamics AX, Microsoft Dynamics GP, Microsoft Dynamics NAV y Microsoft Dynamics SL.

Odoo (conocido anteriormente como OpenERP y anteriormente como TinyERP) - Es un sistema de ERP integrado de código abierto actualmente producido por la empresa belga Odoo S.A. Es un producto de código abierto.

Oracle E-Business Suite - Es uno de los ERP muy potente avalado por ORACLE con un fuerte gestion de base de datos.

¿Qué es SAP?

SAP es una empresa alemana que fue fundada en junio de 1972 las siglas SAP significan "Sistemas, Aplicaciones y Productos en Procesamiento de Datos ". En 1973 SAP empieza con la versión SAP R/1 y no paro desde entonces de mejorar sus sistemas hasta convertirse lo que es hoy en día el líder de los software para empresas a nivel mundial. Se calcula que un 70% de las grandes empresas en el mundo utilizan las aplicaciones de SAP.

¿Qué es ABAP/4?

ABAP/4 es un lenguaje de programación de cuarta generación, su nombre viene de las siglas en Ingles "Advanced Business Application Programming ". Es propiedad de SAP y solo sirve para programar dentro del Sistema SAP ECC las aplicaciones necesarias o mejoras que la empresa necesite. ABAP/4 fue desarrollado para SAP R/2 en los años 80 y no dejo de desarrollarse hasta convertirse en una potente herramienta.

¿Qué es tipos de programas se pueden hacer en ABAP?

ABAP/4 es un lenguaje de programación que permite cuatro principales grupos de programas:

- 1- Reportes clásicos e interactivos.
- 2- Programación de dialogo.
- 3- Batch input
- 4- Programas de comunicaciones

Primeros pasos para conocer la interface de SAP ECC

SAP ECC que es la versión que utilizaremos en este libro para aprender ABAP/4 es un programa cliente servidor, se instala en un servidor de windows con una base de datos de Oracles. A este servidor se le instala diferentes mandantes que no son más que estructuras de datos y procedimiento de una empresa. Algunos mandantes vinan preinstalados en SAP como son:

Mandante 000 Este mandante es el de referencia en el sistema no puede borrarse no cambiar nada en él, no contiene ningún dato de Parametrización.

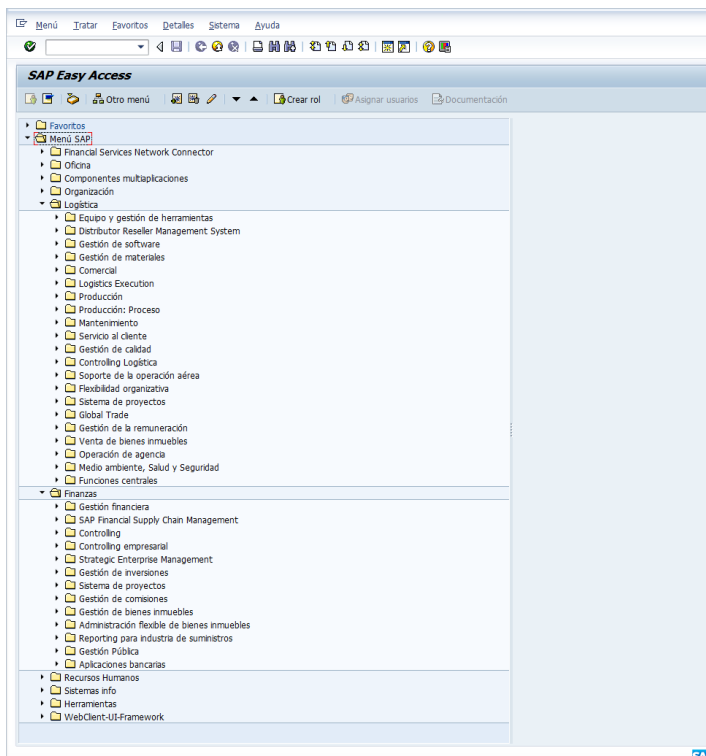
Mandante 001 Este mandante es el de ejemplo se puede borrar o eliminar y la diferencia con 000 es que lo cambiamos nosotros.

Mandante 066 Este es el Mandante del servicio EarlyWatch este mandante no puede ser borrado ni cambiado es utilizado para detención de problemas y nos garantiza la protección de nuestros datos ya que si existe un problema en SAP la empresa se conectará al 066 y no tendrá acceso a nuestros datos empresariales.

Los otros mandantes se pueden crear o pueden venir instalado como es el caso de el mandante 800 en el SAP IDES que por su siglas en Ingles "Internet Demonstration and Evaluation System" que es el SAP de prueba que utiliza SAP para los consultores que necesitan entrenamiento.

Conociendo la interface de SAP

SAP ECC es un ERP que tiene un menú lateral donde se encuentra un árbol con todas las transacciones organizadas por módulos y con una descripción tanto del código como de la función que ejecuta cada una de estas transacciones.

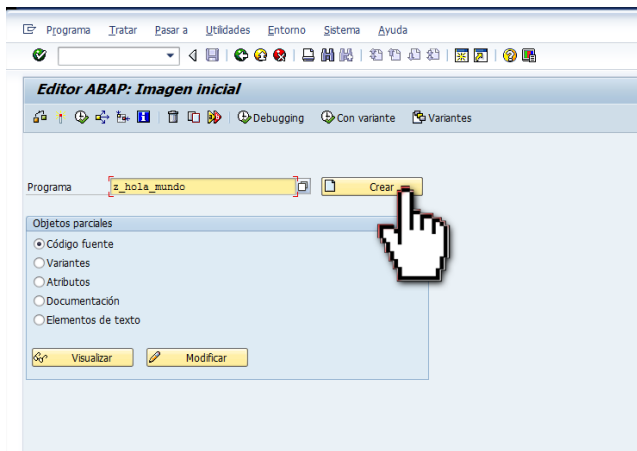


CAPITULO 2 Programación básica en ABAP.

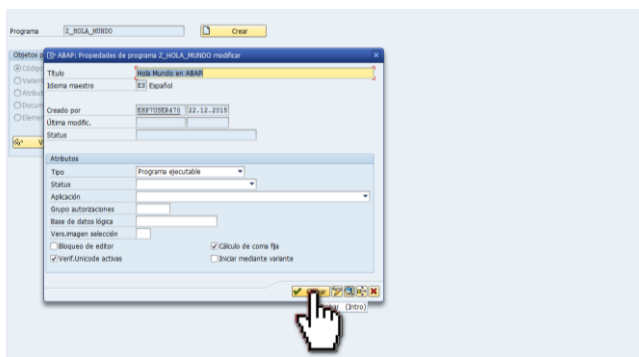
Ejercicio N°1 - Hola Mundo en ABAP/4. (SE38)

En este nuestro primer programa te enseñare como crear un desarrollo z, solamente utilizaremos las función de imprimir en pantalla.

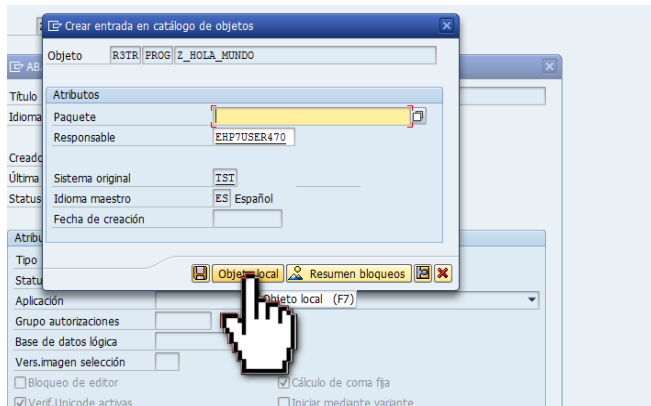
Entramos a la transacción SE38 y en programa colocamos **z_hola_mundo** tal como muestro en la siguiente imagen.



En la próxima ventana en el título le colocamos una descripción corta al programa y pulsamos Crear.

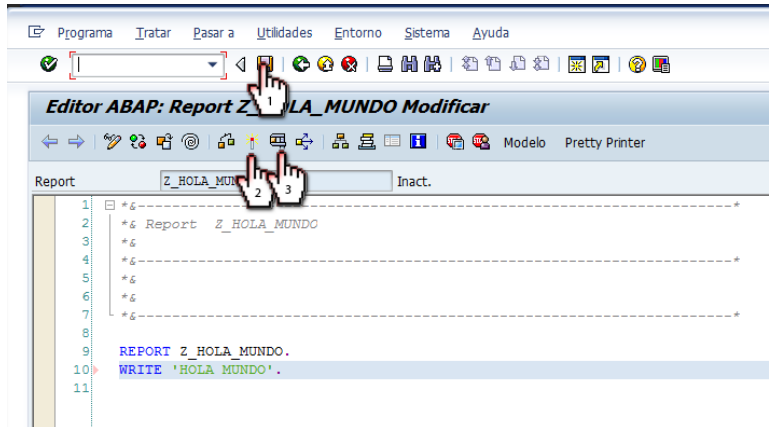


En la próxima ventana pulsamos el botón Objeto Local tal como se muestra.



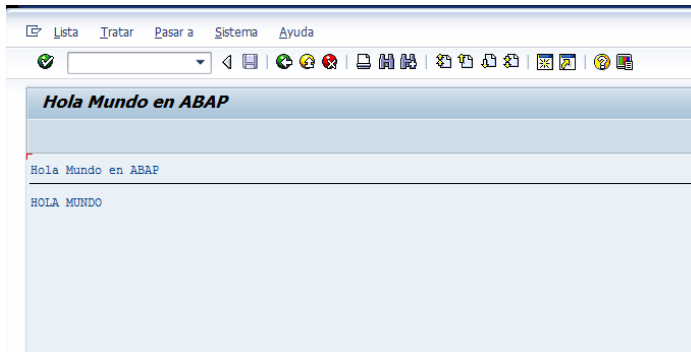
Si todo lo hiciste correctamente veras el editor ABAP donde colocaremos el siguiente código:

```
REPORT Z_HOLA_MUNDO.
WRITE: 'HOLA MUNDO'.
```



Una vez que escribimos el código en el editor ABAP guardamos los cambios en el menú superior, activamos y ejecutamos la aplicación. Nos

saldrá la siguiente ventana, donde veremos el mensaje que acabamos de teclear en el código.



En esta clases aprendimos a crear un programa z, también aprendimos que los programas Z en abap comienzan con la palabra reservada **REPORT** que se utiliza la palabra reservada **WRITE** para imprimir un texto, que todas las líneas en **ABAP/4** terminan con punto y que si queremos imprimir una cadena de caracteres utilizamos comillas simples.

Ejercicio N°2 - Comentarios.

En ABAP/4 los comentarios se pueden hacer de dos formas. Si se hace un comentario desde el inicio del programa se utiliza * si el comentario empieza dentro del texto se utilizas ".

```
REPORT Z_COMENTARIO.  
*Comentario en ABAP  
WRITE: 'HOLA MUNDO'. "Otro comentario en ABAP
```

Ejercicio N°3 - Imprimir varias líneas con WRITE.

Puede ser que en algún programa necesitaras imprimir varias líneas sin tener que repetir la palabra WRITE varias veces, para eso se utiliza la coma para separar las sentencias como se muestra en el ejemplo "Z_IMPRIMIR_3_b".

```
REPORT  Z_IMPRIMIR_3_A.  
WRITE: 'TEXTO 01'.  
WRITE: 'TEXTO 01'.  
WRITE: 'TEXTO 02'.
```

```
REPORT  Z_IMPRIMIR_3_B.  
WRITE: 'TEXTO 01',  
      'TEXTO 01',  
      'TEXTO 02'.
```

Ejercicio N°4 - Tipos de datos en ABAP.

En ABAP/4 tenemos los siguientes tipos de datos primarios:

Tipo	Descripción	Longitud por defecto	Longitud máxima	Valor inicial
C	Alfanuméricos	1	1 - 65535	SPACE
D	Fecha(Date)	8	8	'0000000'
F	Flotante(Float)	8	8	0.0
I	Entero (Integer)	4	4	0
N	Numérico	1	1-65535	'0...0'
P	Empaquetados	8	1-16	0
T	Hora(Time)	6	6	'000000'
X	Hexadecimales	1	1-65535	X'00'

```

REPORT    Z_DATOS_01.
* Declaramos los datos en ABAP con DATA.
DATA: MiEntero01 TYPE I,
DATA: MiEntero02 TYPE I,
DATA: MiEntero03 TYPE I VALUE 12. " Le colocamos 12
* Ahora imprimiremos
WRITE: 'EL NUMERO 3:' , MiEntero03.

```

Ejercicio N°5 - Asignación de valor a una variable

Existen dos formas para asignar una valor a una variable en ABAP, por asignación simple o directa.

```

REPORT    Z_ASIGNACION_SIMPLE.
* Declaramos la variable
DATA: MiEntero01 TYPE I.
* Le asignamos un valor
MiEntero01 = 1.
* Imprimimos el valor
WRITE: MiEntero01.

```

Mediante la sentencia **MOVE**

```

REPORT    Z_ASIGNACION_MOVE.
* Declaramos la variable
DATA: MiEntero01 TYPE I VALUE 123,
      MiEntero02 TYPE I.
* Le asignamos un valor
MOVE MiEntero01 TO MiEntero02.
* Imprimimos el valor
WRITE: MiEntero02.

```

Ejercicio N°6 - Constantes

Las constantes son variables que nunca cambian su valor en toda la ejecución del programa, la definición de VALUE es obligatorio.

```

REPORT    Z_CONSTANTES.
* Declaramos una constante
CONSTANTS: MONEDA(2) TYPE C VALUE 'EUR'.
WRITE: MONEDA.

```

Ejercicio N°7 - Variables del sistema. Imprimir la fecha.

El sistema define un conjunto de variables que son muy útiles para los programadores ABAP/4 entre ellas se encuentra SY-DATUM que nos informa de la fecha del sistema. Si quieres ver otras variables revisa al final del libro la lista de las más utilizada.

```
REPORT    Z_FECHA.
* Imprimimos el valor
WRITE: SY-DATUM.
```

Ejercicio N°8 - Variables del sistema II.

En este ejercicio veremos otras variables del sistemas utilizadas.

```
REPORT    YTEXT.
DATA TEMP TYPE I.
WRITE: / 'EL MANDANTE:', SY-MANDT.
WRITE: / 'USUARIO:', SY-UNAME.
WRITE: / 'IDIOMA:', SY-LANGU.
WRITE: / 'HORA LOCAL:', SY-UZEIT.
WRITE: / 'TRANSACCION:', SY-TCODE.
WRITE: / 'PROGRAMA ACTUAL:', SY-REPID.
DO 10 TIMES.
    TEMP = TEMP + 1.
    " SY-INDEX guarda el contador del LOOP
    IF SY-INDEX = 5.
        WRITE: / 'Es 5'.
    ELSE.
        WRITE: / SY-INDEX.
    ENDIF.
ENDDO.
```

Ejercicio N°9 - Variable SY-SUBRC

Indica que la sentencia anterior a la llamada de esta variable se ejecuto o no.

```
REPORT    Z_SY-SUBRC.
* Seleccionamos la transacción SE11
SELECT SINGLE * tsct WHERE tcode = 'SE11';
```

```
IF SY-SUBRC = 0.
WRITE 'Se encuentra la transacción'.
ENDIF.
```

Ejercicio N°10 - Operaciones con caracteres.

En ABAP a veces necesitamos modificar las salidas de datos formateando las cadenas de textos estas operaciones nos ayudaran con eso.

CONCATENATE: Es utilizado para unir/concatenar campos alfanumérico. Un ejemplo es si queremos unir las variables myvariable01, myvariable02 en la variable myvariable.

```
REPORT Z_OPERACIONES_CONCATENATE.
CONCATENATE myvariable01 myvariable02 INTO
myvariable SEPARATED BY '-'.
* Imprimimos el valor
WRITE: myvariable.
```

CONDENSE: Elimina todos los espacios en blanco en una cadena incluyendo los espacios a la izquierda.

```
REPORT Z_OPERACIONES_CONDENSE.
DATA: VARIABLE TYPE C VALUE ' E U R '.
CONDENSE VARIABLE.
* Imprimimos el valor
WRITE: VARIABLE.
* El valor que imprimirá ser: EUR
```

REPLACE: Se utiliza para reemplazar una parte de la cadena por otra.

```
REPORT Z_OPERACIONES_REMPLACE.
DATA: VARIABLE1 TYPE C VALUE 'MI NOMBRE ES M'.
REPLACE 'M' WITH 'MARLON' INTO VARIABLE1.
* Imprimimos el valor
WRITE: VARIABLE1.
```

SEARCH: Se utiliza para buscar una cadena dentro de otra cadena, si queremos buscar el nombre cuba dentro de una oración si la función encuentra el valor devolverá la variable del sistema SY-SUBRC igual 0, en caso que no la encuentre devolverá 4.

```
REPORT    Z_OPERACIONES_SEARCH .
DATA: ORACION(50) VALUE 'CUB A ES LINDA'.
SEARCH ORACION FOR 'CUB*'
WRITE: /SY-SUBRC. "Imprimirá 0 porque existe.
SEARCH ORACION FOR 'CUBA' ABBREVIATED.
WRITE: /SY-SUBRC. "0:Ignora el espacio vacío.
SEARCH ORACION FOR 'CUBA' STARTING AT 4.
WRITE: /SY-SUBRC. "4:Inicia en la posición 4.
```

SHIFT: Se utiliza para desplazar un conjunto de caracteres eliminando los que no se necesiten.

```
REPORT    Z_OPERACIONES_SHIFT .
DATA: VARIABLE3 TYPE C VALUE 'MI NOMBRE ES M'.
SHIFT VARIABLE3 BY 3 PLACE
* Imprimimos el valor
WRITE: VARIABLE3. "Eliminara los 3 primeros.
```

SPLIT: Se utiliza para partir una cadena en pequeñas partes.

```
REPORT    Z_OPERACIONES_SPLIT .
DATA: VARIABLE1 TYPE C VALUE 'MI,NOMBRE,M'.
SPLIT VARIABLE1 AT ',' INTO VARIABLE11,
VARIABLE12, VARIABLE13.
* Imprimimos el valor
WRITE: VARIABLE11.
```

TRANSLATE: Se utiliza para convertir textos a mayúsculas

```
REPORT    Z_OPERACIONES_SPLIT .
DATA: VARIABLE1 TYPE C VALUE 'hola mundo'.
TRANSLATE VARIABLE1 TO UPPER CASE.
* Imprimimos el valor
WRITE: VARIABLE11.
```

STRLEN: Determina el tamaño de una cadena.

```
REPORT YTEXT.
DATA: A TYPE STRING VALUE 'HOLA MUNDO'.
DATA MY TYPE I.
MY = STRLEN( A ).
WRITE: / MY.
```

Ejercicio N°11 - Ejemplo de uso de SY-SUBRC

En este ejemplo veremos diferentes formas de informar el estado de **SY-SUBRC** la variable interna del sistema que se utiliza para chequear si se ejecuta correctamente un método o función.

```
REPORT Z_SY-SUBRC.
* Imprimirá un texto si la condición es falsa
if sy-subrc <> 0.
    WRITE:/ 'No se encontró'.
endif.
* Mostrará un mensaje si la condición es falsa.
if sy-subrc <> 0.
    MESSAGE 'No se encontró' TYPE 'I'.
endif.
```

Ejercicio N°12 - Ejemplo de uso de líneas tipo tabla.

En este ejemplo veremos cómo usar el comando interno **sy-vline** para crear columnas simples que nos sirven para delimitar un texto en un reporte.

```
REPORT ZCOLOR3.
DATA i TYPE I VALUE 0.
WHILE i < 8.
    WRITE: /'Columna 1', sy-vline,
           'Columna 2', sy-vline.
    i = i + 1.
ENDWHILE.
```


Columna 1	Columna 2	
Columna 1	Columna 2	
Columna 1	Columna 2	
Columna 1	Columna 2	
Columna 1	Columna 2	
Columna 1	Columna 2	
Columna 1	Columna 2	
Columna 1	Columna 2	

Ejercicio N°13 - Ejemplo redondeo varios.

Para redondear un numero tenemos varias opciones como se muestra a continuación.

```
REPORT ZTRANSA.
  DATA N TYPE P DECIMALS 2.
  DATA M TYPE P DECIMALS 2 VALUE '-11.251515'.
* ABS(Absoluto): 5.55
  N = ABS( M ). WRITE: 'ABS: ', N.
* SIGN(Signo): 1.00-
  N = SIGN( M ). WRITE: / 'SIGN: ', N.
* CEIL(Redondeo arriba): 11.00-
  N = CEIL( M ). WRITE: / 'CEIL: ', N.
* FLOOR(Redondeo abajo): 12.00-
  N = FLOOR( M ). WRITE: / 'FLOOR:', N.
* TRUNC(Parte entera): 11.00- (parte entera)
  N = TRUNC( M ). WRITE: / 'TRUNC:', N.
* FRAC(Parte decimal): 0.25-
  N = FRAC( M ). WRITE: / 'FRAC: ', N.
```

Ejercicio N°14 - Operaciones matemática.

Algunas de las operaciones matemáticas más usadas en ABAP.

```
REPORT ZOPERACIONES.
DATA: A TYPE F VALUE 3122.
DATA: B TYPE F VALUE 112.
```

```

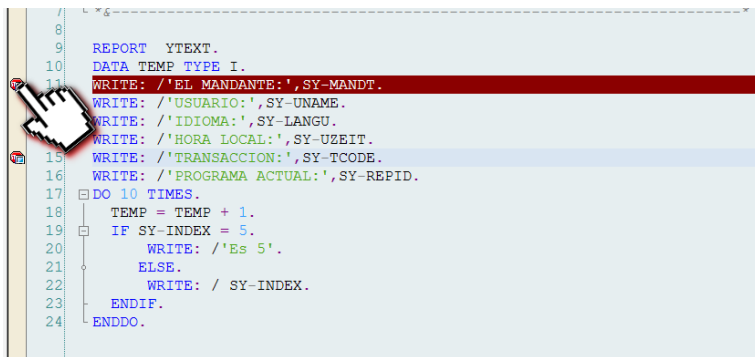
DATA: R TYPE F.
R = A + B. WRITE /: R. " Suma
R = A - B. WRITE /: R. " Resta
R = A / B. WRITE /: R. " Division
R = A * B. WRITE /: R. " Multiplicacion
R = A DIV B. WRITE /: R. " Division Entera
R = SIN( A ). WRITE /: R. " Seno
R = COS( B ). WRITE /: R. " Coseno
R = SQRT( B ). WRITE /: R. " Raiz
R = LOG( B ). WRITE /: R. " logaritmo neperiano
R = LOG10( B ). WRITE /: R. " logaritmo 10

```

Ejercicio Nº15 - ABAP Debugger.

Si queremos revisar el código paso a paso del programa para ver que valores toman las variables utilizamos el ABAP Debugger que es una herramienta de verificación muy importante que trae el sistema. Para activarla usaremos estos paso.

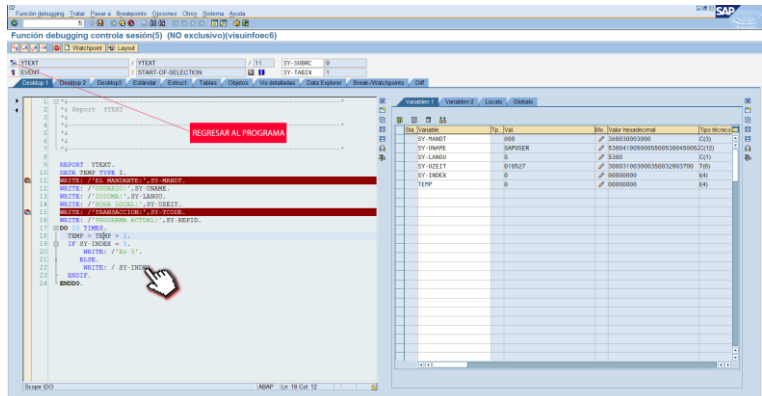
Paso 1: Marcamos los **Breakpoint** en el programa dando un click sobre la línea donde queremos que el programa haga una pausa y lo ejecutamos con [F8].



Si pulsamos [F5] iremos paso a paso comprobando los valores de las variables, para conocer estos valores hay dos formas:

- 1- Parándonos con el **Mouse** sobre la variable.

2- Dando doble click izquierdo las variables pasarían a la ventana de la derecha y se actualizarían en cada paso de la ejecución del programa.



Si queremos salir del modo de debugger pulsamos Menu superior [Pasar a] [Navegar a código fuente].

CAPITULO 3 Operadores de condición.

Ejercicio N°16 - Sentencia CHECK

La sentencia **CHECK** se utiliza para saber si una condición es verdadera

```
REPORT  Z_COND_CHECK .
* Comprobamos que la variable SY-SURC = 0.
CHECK SY-SUBRC EQ 0.
WRITE: VARIABLE11.
```

Ejercicio N°17 - Sentencia IF ..ELSE

La sentencia **IF..ELSE** es una de las más utilizadas en ABAP/4 con ella se compara un valor si cumple una condición ejecuta un código.

```
REPORT  Z_IF_ESLE .
DATA: AUTO1(10) TYPE C VALUE 'CAMARO',
      AUTO2(10) TYPE C VALUE 'CRUZE'.
IF AUTO1 = 'CAMARO'.
  WRITE: 'MI AUTO FAVORITO ES UN CAMARO'.
ELSEIF AUTO1 = 'CRUZE'.
  WRITE: 'MI AUTO FAVORITO ES UN CRUZE'.
ELSE: 'MI AUTO ACTUAL ES UN CRUZE'.
ENDIF.
```

Listados de operadores lógicos.

Operador	Operador	Descripción
=	EQ	Es igual
>	GT	Mayor que
<	LT	Menor que
>=	GE	Mayor o igual que
<=	LE	Menor o igual que
<>	NE	Diferente
BETWEEN Valor1 and Valor 2		Entre
IS INITIAL		El contenido no ha cambiado
IS NOT INITIAL		El contenido ha cambiado

Ejercicio N°18- Sentencia CASE

La sentencia **CASE** se utiliza cuando una variable tiene múltiples opciones y a cada una de las opciones se le quiere dar una función.

```
REPORT  Z_CASE.
DATA: MiEntero01 TYPE I VALUE 1.
CASE MiEntero01
  WHEN 1:
    WRITE: 'MI AUTO FAVORITO ES UN CAMARO'.
  WHEN 2:
    WRITE: 'MI AUTO FAVORITO ES UN CRUZE'.
ENCASE.
```

Ejercicio N°19- Usando el operador <> diferente.

El operador <> se utiliza cuando queremos comparar dos valores para conocer si son diferentes.

```
REPORT    Z_DIFERENTE .
DATA: MiEntero01 TYPE I VALUE 1,
      MiEntero02 TYPE I VALUE 2.
IF MiEntero01 <> MiEntero02
    WRITE: 'Los números no son iguales'.
ELSE
    WRITE: 'Los números son iguales'.
ENIF.
```

Ejercicio N°20- Usando el BETWEEN

El operador **BETWEEN** se utiliza cuando queremos saber si una variable se encuentra en un rango.

```
REPORT    Z_BETWEEN .
DATA: MiEntero01 TYPE I VALUE 1.
IF MiEntero01 BETWEEN 0 AND 9
    WRITE: 'Mi entero está entre el 1..9'.
ENIF.
```

CAPITULO 3 Bucles.

Ejercicio N°21 - Bucle DO

El bucle **Do** se utiliza para ejecutar una función mientras esta cumple una condición.

```
REPORT  Z_DO_E15.  
DATA: Entero TYPE I VALUE 1.  
DO 10 TIMES  
    Entero = Entero + 1.  
    WRITE: /'El numero es'.Entero.  
ENDDO.
```

Ejercicio N°22 - Bucle DO con EXIT

El bucle **Do** a veces se combina con un EXIT para salir del bucle si determinada condición se cumple.

```
REPORT  Z_DO_E16.  
DATA: Entero TYPE I VALUE 1.  
DO 10 TIMES  
    IF Entero = 5.  
        EXIT.  
    Entero = Entero + 1.  
    WRITE: /'El numero es'.Entero.  
ENDDO.
```

Ejercicio N°23 - Bucle While

El bucle **While** se utiliza para ejecutar una función mientras se cumpla una condición.

```
REPORT  Z_WHILE_E17.  
DATA: Entero TYPE I VALUE 1.  
WHILE Entero LT 10
```

```
Entero = Entero + 1.  
WRITE: /'El numero es'.Entero.  
ENWHILE.
```


CAPITULO 4 Parámetros y subrutinas.

Ejercicio N°24 - Parámetros de entradas.

En ocasiones necesitamos entrar un valor para procesarlo para eso se utiliza los parámetros de entradas que se nombran con la palabra reservada PARAMETERS.

```
REPORT Z_PARAMETROS_E18.  
PARAMETERS: Numero1 TYPE I.  
PARAMETERS: Numero2 TYPE I.  
DATA: RESULTADO TYPE I.  
RESULTADO = Numero1 + Numero2.  
WRITE: RESULTADO.
```

Ejercicio N°25 - Subrutinas Internas.

Las subrutinas son utilizadas para fragmentar el código de tal forma que sea mucho más fácil su mantenimiento, para crear una subrutina lo hacemos con la palabra reservada PERFORM.

```
REPORT Z_SUBROUTINAS_E19.  
PERFORM MSUBROUTINA.  
* Implementación de la subrutina.  
FORM MSUBROUTINA.  
WRITE: 'Esto es un ejemplo de subrutina  
interna'.  
ENDFORM.
```

Ejercicio N°26 - Subrutinas Internas con parámetros.

Las subrutinas permiten pasar parámetros igual que las funciones.

```
REPORT Z_SUBROUTINAS_E19.  
DATA: Entero TYPE I VALUE 1.  
PERFORM MSUBROUTINA2 USING ENTERO.  
* Implementación de la subrutina.  
FORM MSUBROUTINA2 USING ENTERO.  
WRITE: 'Imprimira el Entero:', ENTERO.  
ENDFORM.
```

Ejercicio N°27 - Subrutinas Externa con parámetros.

Las subrutinas también pueden estar ubicadas en ficheros externos para hacer el trabajo más organizado es una de la buenas prácticas de programación ABAP/4.

1 . Creamos el primer programa que le llamaremos Z_DB

```
REPORT DB.
DATA: Entero TYPE I VALUE 1.
PERFORM MSUBRUTINA3 USING ENTERO.
* Implementación de la subrutina.
FORM MSUBRUTINA3 USING ENTERO.
WRITE: 'Imprimira el Entero:', ENTERO.
ENDFORM.
```

2 . Creamos el segundo programa Z_CALCULO que llamará la subrutina

```
REPORT Z_CALCULO.
PERFORM MSUBRUTINA3 USING ENTERO(Z_DB) USING
ENTERO.
```

Ejercicio N°28 - Plantilla de ABAP/4

Es importante en la programación mantener un orden para que el código sea legible por cualquier otro programados para eso se utiliza las plantillas. Esto es un ejemplo de una plantilla muy simple.

```
*&-----
*& Nombre Reporte: Z_PLANTILLA_E22
*& Autor: MARLON
*& FECHA: 26.12.15
*&-----
*& Modificaciones: Se agregaron los Datos
*& Fecha: 26.12.15
*&-----
REPORT Z_PLANTILLA_E22.
*****
* DATOS
```

```
*****  
DATA: Entero TYPE I VALUE 1.  
*****  
* SALIDA DE DATOS  
*****  
WRITE: 'Imprimirá'.
```

CAPITULO 5 Tablas internas.

Ejercicio N°29 - Tablas internas.

En ABAP/4 a veces necesitamos procesar varios registros en tiempo de ejecución, para eso se utilizan las tablas internas, son estructuras de base de datos que pueden tener la misma estructura de la tabla fuente o pueden tener estructura propia. La palabra reservada **OCCURS** especifica la cantidad de registro que se guarda en la memoria, cuando el valor es 0 significa que puede tener cualquier cantidad de registros.

```
REPORT  Tablas_Internas_E23 .
* Declaramos una tabla interna
DATA: BEGIN OF INT_ALUMNOS OCCURS 0,
      Nombre(25)      TYPE C,
      Apellidos(25)   TYPE C,
      EDAD(3)         TYPE I,
      FECHA           TYPE D,
END OF INT_ALUMNOS.
* Llenamos la tabla interna
TI_ALUMNOS-Nombre = 'Marlon'.
TI_ALUMNOS-Apellidos = 'Falcon'.
TI_ALUMNOS-EDAD = '35'.
TI_ALUMNOS-FECHA= SY-DATUM.
* Actualizamos los datos
APPEND TI_ALUMNOS
* Recorremos la tabla interna mediante LOOP
LOOP AT TI_ALUMNOS.
  WRITE: / INT_ALUMNOS-NOMBRE, INT_ALUMNOS-
Apellidos.
ENDLOOP.
```

Ejercicio N°30 - Tablas internas estructura general.

Esta es la estructura general de una tabla interna.

```
REPORT  ZEJEMPLO1.
TABLES: T1,T2.
DATA: BEGIN OF <Nombre de la tabla interna>.
```

```

CAMPO01 TYPE I,
CAMPO02 TYPE C.
END OF INTERNAL <Nombre de la tabla interna>.

```

En el nombre de la tabla interna se coloca cualquier cadena de carácter pero como patrón siempre es bueno empezar con **INT_*** es decir si la tabla se llama **CASA** la tabla interna se llamará **INT_CASA**. En el siguiente ejemplo veremos cómo adicionar registros a una tabla interna.

```

REPORT  Tablas_Internas_E24.
TABLES: Zusuuario.
DATA: BEGIN OF F_CADENA.
      nombre(13)      TYPE C.
      apellidos(13)   TYPE C.
      mail(20)        TYPE C.
DATA: END OF F_CADENA.
DATA: BEGIN OF INT_TABLA OCCURS 200.
      nombre(13)      TYPE C.
      apellidos(13)   TYPE C.
      mail(20)        TYPE C.
DATA: END OF INT_TABLA.
Zusuuario-nombre = "MARTHA".
Zusuuario- apellidos = "HERNANDEZ".
Zusuuario- mail = "a@123marlon.cl".
START-OF-SELECTION.
      SELECT * FROM Zusuuario WHERE Apellidos LIKE
      'H%'.
      MOVE Zusuuario-nombre TO INT_TABLA-nombre.
      MOVE Zusuuario-apellidos TO INT_TABLA-
      apellidos.
      MOVE Zusuuario-mail TO INT_TABLA-mail.
      CLEAR F_CADENA.
      ENDSELECT.
      LOOP AT INT_TABLA.
          WRITE: / INT_TABLA-nombre.
      ENDLOOP.

```

Ejercicio N°31 - Tablas internas con cabecera.

Si bien se puede no incluir la cabecera de la tabla interna siempre es recomendable colocarla, en este ejemplo veremos cómo hacerlo. Como

concepto la cabecera es el único registro que se utiliza para agregar y recorrer datos del cuerpo de la tabla interna.

```
REPORT  Z_TABLAS_INTERNAS_CON_E25.
* Creamos la estructura de la tabla interna.
* utilizamos la tabla KNA1 maestro de clientes.
TYPES: BEGIN OF st_KNA1,
        NAME1    LIKE KNA1-NAME1, " Nombre
        TELF1    LIKE KNA1-TELF1, " Telefono
        LAND1    LIKE KNA1-LAND1, " Pais
      END OF st_KNA1.
* Creamos la tabla interna INT_KNA1.
DATA: INT_KNA1 TYPE STANDARD TABLE OF st_KNA1
WITH HEADER LINE.
* Campo para que el usuario entre un valor
* US-Partner
PARAMETERS: p_NAME1 LIKE KNA1-NAME1.
* Llenamos la tabla
SELECT NAME1 TELF1 LAND1
      FROM KNA1 INTO TABLE INT_KNA1
      WHERE NAME1 = p_NAME1.
*Imprimimos los datos de la tabla interna
LOOP AT INT_KNA1.
    WRITE: / INT_KNA1-NAME1,
            INT_KNA1-TELF1,
            INT_KNA1-LAND1.
ENDLOOP.
```

Ejercicio N°32 - Tablas internas consulta de usuarios.

En este ejemplo veremos cómo importar la estructura de una tabla de SAP, el ejemplo muestra el nombre, mandante y ultima fecha de entrada al sistema de un usuario en SAP. Para eso utilizaremos la tabla USR02.

```
*&-----*
*& REPORT  ZUSER12
*& Autor  MARLON FALCON
*&-----*
REPORT  ZUSER12.
* Creamos la estructura de la tabla USR02
TABLES: USR02.
DATA: BEGIN OF ST_USUARIOS.
```

```

        INCLUDE STRUCTURE USR02.
DATA END OF ST_USUARIOS.
* Creamos la tabla interna con cabecera
DATA INT_USUARIOS LIKE TABLE OF ST_USUARIOS WITH
HEADER LINE.
* Entramos el Usuario que queremos consultar
PARAMETERS: PUSUARIO LIKE USR02-BNAME.
* Hacemos una consulta a la tabla USR01 y llenamos
la tabla interna.
* Imprimimos en pantalla los valores.
SELECT * FROM USR02 INTO CORRESPONDING FIELDS OF
TABLE INT_USUARIOS
        WHERE BNAME = PUSUARIO.
LOOP AT INT_USUARIOS.
    WRITE: / INT_USUARIOS-BNAME,
            INT_USUARIOS-MANDT,
            INT_USUARIOS-TRDAT.
ENDLOOP.

```

Ejercicio N°33 - Tabla interna simple.

Este es un ejemplo simplificado de un tabla interna.

```

REPORT ZSQL1.
TABLES: MARAV.
DATA: INT_MARAV LIKE MARAV OCCURS 1000 WITH HEADER LINE.
SELECT * FROM MARAV INTO TABLE INT_MARAV.
LOOP AT INT_MARAV.
    WRITE:/ INT_MARAV-MANDT.
ENDLOOP.

```

Ejercicio N°34 - Tablas internas con Áreas de trabajo.

Este es un ejemplo veremos cómo trabajar con áreas de trabajos y tablas sin cabecera. En el ejemplo se mostrará el primer registro de una tabla.

```

REPORT ZOPERACIONES.
* Declaro el área de trabajo.
DATA WA_ICON LIKE ICON.
* Declaro el parametro de entrada.

```

```
PARAMETER PA_NAME LIKE ICON-NAME DEFAULT 'ICON_TOTAL_LEFT'.
```

** Selecciono el primer valor*

```
SELECT SINGLE NAME ID
```

```
FROM ICON INTO CORRESPONDING FIELDS OF WA_ICON
```

```
WHERE NAME = PA_NAME.
```

```
WRITE: / WA_ICON-ID,
```

```
WA_ICON-NAME COLOR COL_KEY.
```

Ejercicio N°35 - Tablas internas con FIELD-SYMBOLS

Este es un ejemplo veremos cómo trabajar con **FIELD-SYMBOLS** que no son más que punteros en ABAP. Un puntero se conoce como una variable que hace referencia a una dirección de memoria en un programa.

```
REPORT ZOPERACIONES.
```

** Creamos la estructura de la tabla.*

```
TYPES: BEGIN OF ST_ICON,
```

```
    ID LIKE ICON-ID,
```

```
    NAME LIKE ICON-NAME,
```

```
END OF ST_ICON.
```

** Creamos el parámetro de entrada.*

```
PARAMETER PA_NAME like ICON-NAME DEFAULT 'ICON_TOTAL_LEFT'.
```

** Declaro la tabla interna si cabecera*

```
DATA: TI_ICON TYPE STANDARD TABLE OF ST_ICON.
```

** Declaro el FIELD-SYMBOLS*

```
FIELD-SYMBOLS: <FS_ICON> LIKE LINE OF TI_ICON.
```

** Hacemos la consulta*

```
SELECT NAME ID
```

```
FROM ICON INTO TABLE TI_ICON
```

```
WHERE NAME = PA_NAME.
```

** Recorremos la tabla interna*

```
LOOP AT TI_ICON ASSIGNING <FS_ICON>.
```

```
    WRITE: / <FS_ICON>-ID,
```

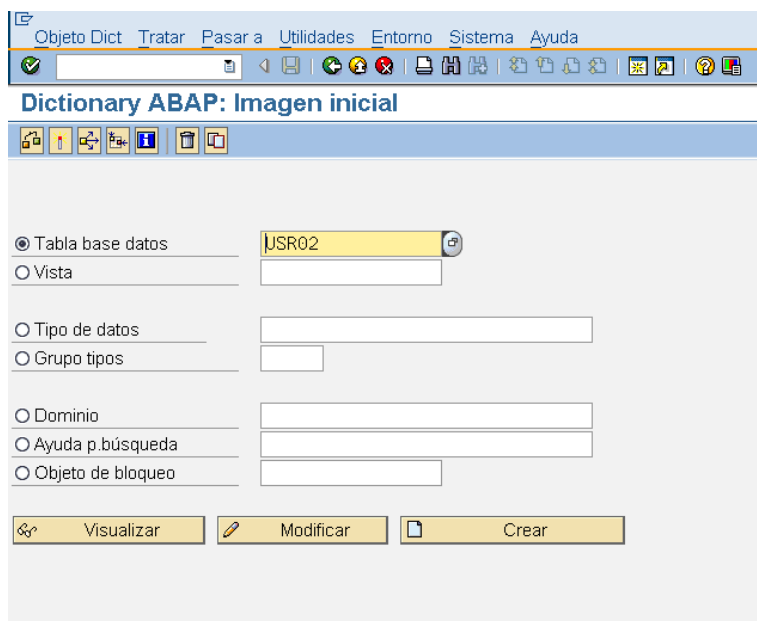
```
           <FS_ICON>-NAME.
```

```
ENDLOOP.
```


CAPITULO 5 Diccionario de datos

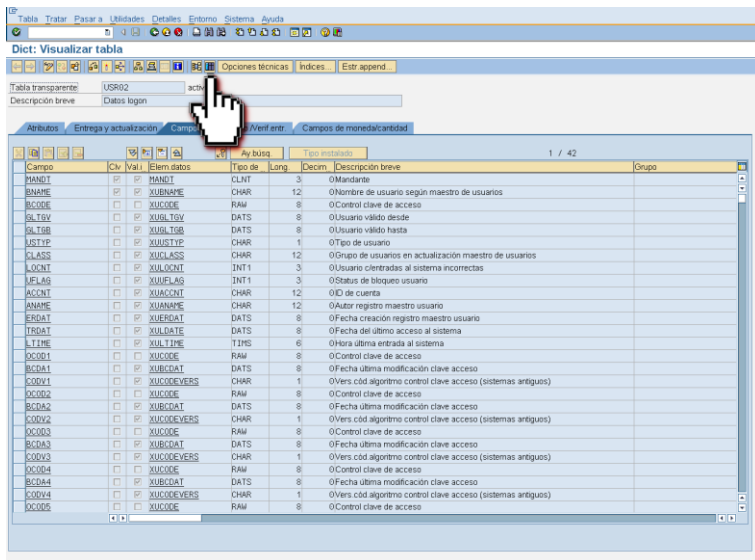
Ejercicio N°36 - Entrando al DD (SE11)

En el diccionario de datos están todas las tablas que describen el sistema SAP. Para entrar el diccionario lo hacemos mediante el **MENU/HERRAMIENTAS/WORKBENCH ABAP/DESARROLLO/SE11 DICTIONARY ABAP** o también si vamos directo a la transacción **SE11**.

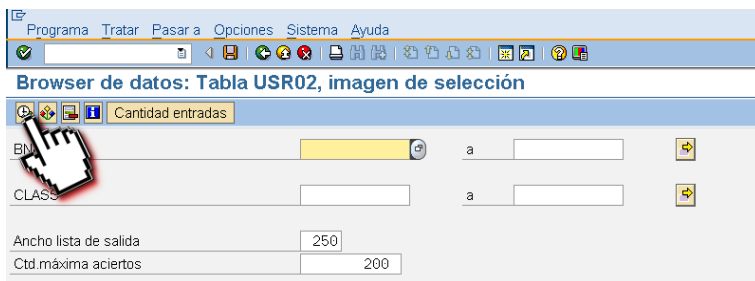


En el campo "Tabla de base datos" entramos la tabla que queremos consultar en este ejemplo usaremos la **USR02** que muestra el listado de usuarios en SAP. Una vez seleccionada la tabla pulsamos el botón inferior Visualizar. Y veremos la estructura de la tabla

consultada. Para visualizar los Datos tenemos que pulsar la combinación de teclas **Ctrl + Shift + F10** o pulsamos el botón **Contenido**.



Nos saldrá una ventana de y pulsamos "F8" o el botón ejecutar.








Una ejecutada nos mostrará todos los registros de la tabla consultada USR02.

[illegible]

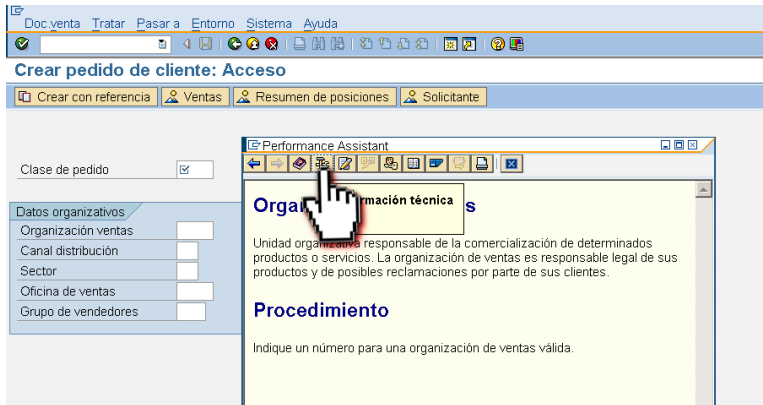
Ejercicio N°37 - Tabla de una transacción.

Las transacciones en SAP están compuestas por diferentes datos, para saber de dónde SAP toma los datos tiene que seguir estos pasos.

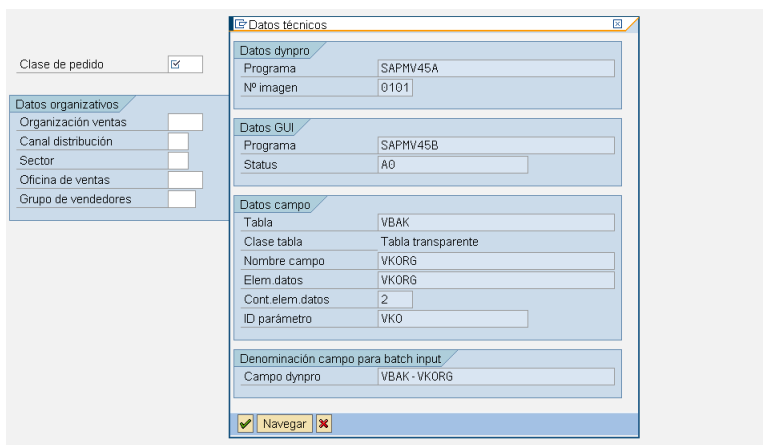
1- Ejecutamos la transacción que queremos revisar en este ejemplo utilizare la VA01 - Crear oferta de venta. Sobre el campo organización de venta presionamos la tecla F1 "Ayuda".

Clase de pedido	<input checked="" type="checkbox"/>
Datos organizativos	
Organización ventas	
Canal distribución	
Sector	
Oficina de ventas	
Grupo de vendedores	

2- En la ventana de ayuda pulsamos Información técnica.



3- En la ventana **datos técnicos** fíjate que te muestra el nombre de la tabla y el campo, en este caso la tabla es la **VBAK**. Ahora si vas a las **SE11** y buscas la tablas veras todos los datos que puedes consultarle desde ABAP.



Ejercicio N°38 - Consulta para una tabla de DD.

Esta plantilla te servirá para una vez que tengas una tabla del DD (Diccionario de Datos) y quieras consultar todo su contenido desde ABAP lo puedas hacer, usaremos la misma tabla del ejemplo anterior.

```
*&-----*
*& Report ZOFERTAS
*& Autor MARLON FALCON
*&-----*
REPORT ZOFERTAS.
* Creamos la estructura de la tabla interna con la tabla VBAK
TABLES: VBAK.
DATA: BEGIN OF ST_VBAK.
    INCLUDE STRUCTURE VBAK.
DATA END OF ST_VBAK.
* Creamos la tabla interna con cabecera
DATA INT_VBAK LIKE TABLE OF ST_VBAK WITH HEADER LINE.
* Imprimimos en pantalla los valores.
SELECT * FROM VBAK INTO CORRESPONDING FIELDS OF TABLE INT_VBAK.
LOOP AT INT_VBAK.
    WRITE: / INT_VBAK-VKORG, " Organización de Venta
           INT_VBAK-VTWEG, " Canal de distribución
           INT_VBAK-SPART. " Sector
ENDLOOP.
```

Ejercicio N°39 - Reporte AVL con una tabla del DD.

Esta plantilla te servirá para una vez que tengas una tabla del DD (Diccionario de Datos) y quieras consultar mediante un Reporte ALV

```
REPORT ZALV01.
* Cargamos los datos tipo slis
TYPE-POOLS SLIS.
DATA: G_INT_fieldcat TYPE slis_t_fieldcat_alv,
      G_ST_fieldcat TYPE slis_fieldcat_alv.
* Creamos la estructura de la tabla interna con
  la tabla MARAV
TABLES: MARAV.
DATA: BEGIN OF ST_MARAV.
    INCLUDE STRUCTURE MARAV.
DATA END OF ST_MARAV.
* Creamos la tabla interna con cabecera
```

```

DATA INT_MARAV LIKE TABLE OF ST_MARAV WITH
  HEADER LINE.
* Copiamos los datos tabla MARAV a la tabla
  interna INT_MARAV
SELECT * FROM MARAV INTO TABLE INT_MARAV.
* Imprimimos en pantalla los valores. Fieldcat
G_ST_fieldcat-fieldname = 'MANDT'. "Nombre del
  campo de la tabla
G_ST_fieldcat-seltext_m = 'Mandante'.
  "Descripción mediana
G_ST_fieldcat-seltext_s = 'Man.'. "Descripción
  corta
APPEND G_ST_fieldcat TO G_INT_fieldcat.

G_ST_fieldcat-fieldname = 'MTART'.
G_ST_fieldcat-seltext_m = 'Tipo de material'.
G_ST_fieldcat-seltext_s = 'Tip.'.
APPEND G_ST_fieldcat TO G_INT_fieldcat.
* Función para mostrar el ALV
CALL FUNCTION 'REUSE_ALV_GRID_DISPLAY'
  EXPORTING
    it_fieldcat    = G_INT_fieldcat[]
    I_GRID_TITLE   = 'ZALV01/Tabla-vista para BD
      lógica MGM'
  TABLES
    t_outtab       = INT_MARAV[]
  EXCEPTIONS
    program_error  = 1
    OTHERS         = 2.

```

Ejercicio N°40 - Full Reporte AVL de una tabla.

Esta plantilla te servirá para una vez que tengas una tabla del DD (Diccionario de Datos) y quieras consultar mediante un Reporte ALV.

```

REPORT ZALV04.
* Cargamos los datos tipo slis
TYPE-POOLS SLIS.
DATA: G_INT_fieldcat TYPE slis_t_fieldcat_alv,
      G_ST_fieldcat TYPE slis_fieldcat_alv.
* Creamos la estructura de la tabla MARAV

```

```

TABLES: MARAV.
DATA: BEGIN OF ST_MARAV.
    INCLUDE STRUCTURE MARAV.
DATA END OF ST_MARAV.
* Creamos la tabla interna con cabecera
DATA INT_MARAV LIKE TABLE OF ST_MARAV WITH
HEADER LINE.
* Copiamos los datos tabla MARAV a la tabla
interna INT_MARAV
SELECT * FROM MARAV INTO TABLE INT_MARAV UP TO
100 ROWS.
* Llamamos la función para saber todos los
campos de la tabla.
DATA: BEGIN OF INT_TAB OCCURS 100.
    INCLUDE STRUCTURE DFIES.
DATA: END OF INT_TAB.
    call function 'DDIF_FIELDINFO_GET'
        exporting
            tabname                = 'MARAV'
*            FIELDNAME              = ' '
            LANGU                  = SY-LANGU
*            LFIELDNAME             = ' '
*            ALL_TYPES              = ' '
*            IMPORTING
*            X030L_WA               = WATAB
*            DDOBJTYPE              =
*            DFIES_WA               =
*            LINES_DESCR            =
        TABLES
            DFIES_TAB               = INT_TAB
*            FIXED_VALUES           =
            EXCEPTIONS
            NOT_FOUND               = 1
            INTERNAL_ERROR          = 2
            OTHERS                  = 3.
if sy-subrc <> 0.
    WRITE:/ 'No se encuentra los campos'.
endif.
* Imprimimos en pantalla los valores. Fieldcat
LOOP AT INT_TAB.
G_ST_fieldcat-fieldname = INT_TAB-FIELDNAME.
G_ST_fieldcat-seltext_m = INT_TAB-FIELDTEXT.

```



```
G_ST_fieldcat-seltext_s = INT_TAB-FIELDNAME.
APPEND G_ST_fieldcat TO G_INT_fieldcat.
ENDLOOP.
```

```
* Función para mostrar el ALV
CALL FUNCTION 'REUSE_ALV_GRID_DISPLAY'
  EXPORTING
    it_fieldcat    = G_INT_fieldcat[]
  I_GRID_TITLE    = 'ZALV01/Titulo ALV'
  TABLES
    t_outtab       = INT_MARAV[]
  EXCEPTIONS
    program_error  = 1
    OTHERS         = 2.
```

Ejercicio N°41 - Crear un Dominio.

El dominio en SAP se le llama a el rango de valores que puede tener un campo de la tabla. Ejemplo el dominio : **COLOR_AUTO** puede tener (Rojo, Verde, Azul, etc.). El dominio nos guarda información de tamaño máximo, tipo de información que le colocaremos. Para crear un dominio iremos a las transacción **SE11** y crearemos el dominio **ZD_COLOR_AUTO**.

☐ Tabla base datos
☐ Vista
☐ Tipo de datos
☐ Grupo tipos
☒ Dominio
☐ Ayuda p.búsqueda
☐ Objeto de bloqueo

ZMITABLA

ZD_COLOR_AUTO

Visualizar Modificar Crear

Crear

Entramos las variables:

Descripción breve: Podemos poner un texto que describa el campo.

Tipo de datos: Seleccionamos CHAR para decir que es una cadena.

Ctd. posiciones: 10

Longitud de salida: 10

Dominio: ZD_COLOR_AUTO nuevo(revisado)

Descripción breve: Colores de autos chile

Definición

Formato

Tipo de datos: CHAR String

Ctd. posiciones: 10

Decimales:

Propiedades salida

Longitud salida: 10

Rutina conv.:

☐ Signo +/-

☐ Minúsculas

Vamos a la pestaña Ámbitos val y definimos los valores que puede tomar el campo que este caso son los posibles colores.

Dict: Actualizar dominios

Dominio: ZD_COLOR_AUTO nuevo(revisado)

Descripción breve: Colores de autos chile

Ámbito val.

Val. fijo	Descrip. breve
Verde	Color 1
Azul	Color 2
Roj o	Color 3
Amarillo	Color 4

Por último Guardamos, damos una orden de transporte y activamos.

Ejercicio N°42 - Crear un Elemento de dato.

Los elementos de datos contienen el conjunto de la descripción del campo y el dominio. Para crearla lo haremos también con la transacción SE11. En tipo de datos colocamos "ZED_COLOR_AUTO".

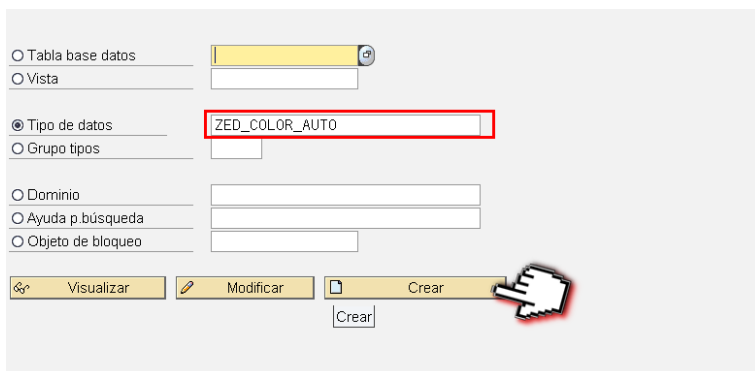


Diagrama de la interfaz de usuario para crear un elemento de dato en SE11:

- Radio buttons: ☐ Tabla base datos, ☐ Vista, ☒ Tipo de datos, ☐ Grupo tipos.
- Campos de texto: (destacado con un recuadro rojo).
- Radio buttons: ☐ Dominio, ☐ Ayuda p.búsqueda, ☐ Objeto de bloqueo.
- Botones: Visualizar, Modificar, Crear (con un cursor de mouse encima).
- Botón secundario: Crear.

En la próxima ventana seleccionamos Elem.datos y pulsamos continuar.

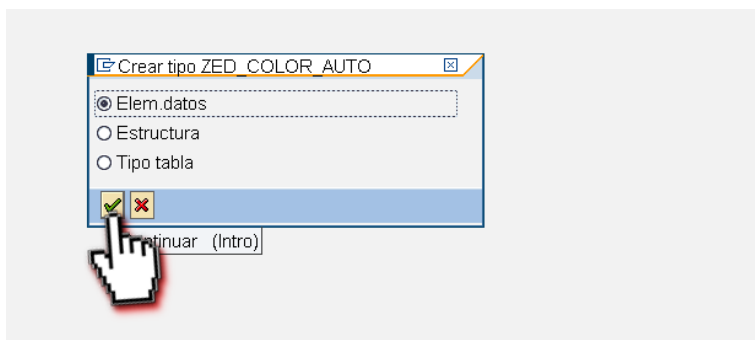


Diagrama de la interfaz de usuario para seleccionar el tipo de elemento de dato:

- Título: Crear tipo ZED_COLOR_AUTO.
- Radio buttons: ☒ Elem.datos, ☐ Estructura, ☐ Tipo tabla.
- Botones: Continuar (Intro) (con un cursor de mouse encima).

Creamos una descripción breve del elemento de dato, colocamos el dominio que queremos que contenga.

Elemento datos: ZED_COLOR_AUTO nuevo(revisado)

Descripción breve: COLOR DE AUTO

Tipo datos

☒ Tipo elemental

☒ Dominio

ZD_COLOR_AUTO

Tipo datos: ZD_COLOR_AUTO

Longitud: 0 Decimales: 0

☐ Tipo instalado

Tipo datos:

Longitud: 0 Decimales: 0

☐ Tipo referencia

☐ Tipo referenciado

☐ Referencia a tipo instalado

Tp. datos:

Longitud: 0 Decimales: 0

En la pestaña **Denom.campo** colocaremos las diferentes descripción que aparecerá en la tabla en dependencia del tamaño que se muestre. Después guardamos y activamos.

Elem. datos Tratar Pasar a Utilidades Entorno Sistema Ayuda

Dict: Actualizar elemento

Elemento datos: ZED_COLOR_AUTO nuevo(revisado)

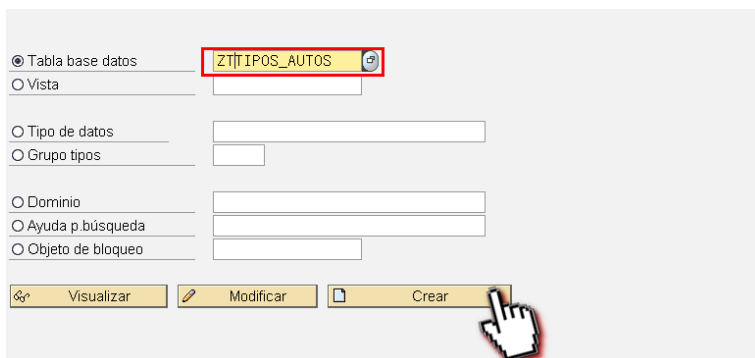
Descripción breve: COLOR DE AUTO

Denom.campo

	Long.	Denominador de campo
Breve	3	COL
Mediano	6	COLOR
Largo	15	COLOR DE AUTO
Cabecera	5	COLOR

Ejercicio N°43 - Crear tabla transparente (SE11).

Para crear una tabla transparente entraremos a la transacción SE11 y con Tabla de base de datos seleccionado entramos el nombre de ejemplo que usaremos que es **ZTTIPOS_AUTOS** y pulsamos el botón crear.

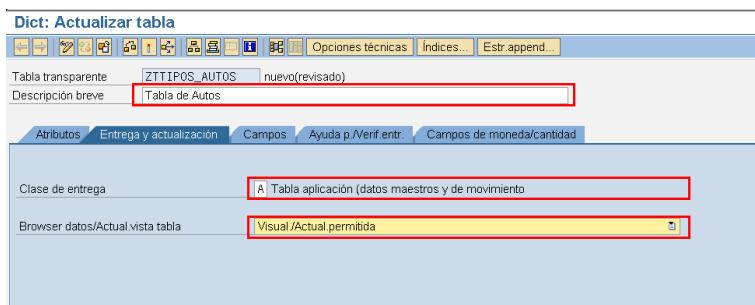


Colocamos los siguientes datos:

Descripción breve: Descripción de la tabla

Clase de entrega: A "Porque almacenara datos, tabla de aplicación.

Browser datos: Visual./Actual.permitida.



ID: pulsamos el botón TIPOS INSTALADOS y buscamos INT1

MODELO: buscamos el elemento de datos "ALV_CHAR40".

COLOR: buscamos el elemento de datos que creamos que se llama "ZED_COLOR_AUTO".

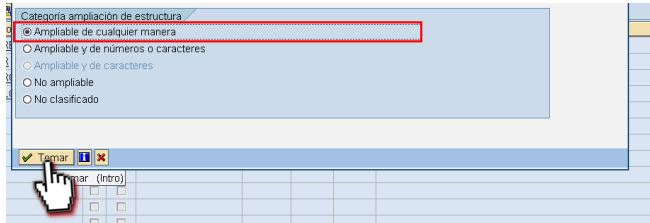
Campo	Clv	Val i	Elem. datos	Tipo de ...	Long.	Decim.	Descripción breve
ID	<input checked="" type="checkbox"/>	<input type="checkbox"/>		INT1	3	0	
COLOR	<input type="checkbox"/>	<input type="checkbox"/>	ZED_COLOR_AUTO	CHAR	10	0	COLOR DE AUTO
MODELO	<input type="checkbox"/>	<input type="checkbox"/>	ALV_CHAR40	CHAR	40	0	Campo texto 40

Defino cual será el valor clave en este caso utilizaremos siempre la primera fila de la tabla por lo que en la columna **Clv** la marco. Guardamos y en la próxima ventana cuando nos pida la Clase de datos colocamos APPL0 que significa datos maestros, **Categ.tamaño** ponemos 0 que quiere decir datos entre 0...6000 registros.

Parámetros memoria lógicos	
Clase de datos	APPL0 Datos maestros, tablas transparentes
Categ.tamaño	0 Reg. datos esperados 0 a 6.000

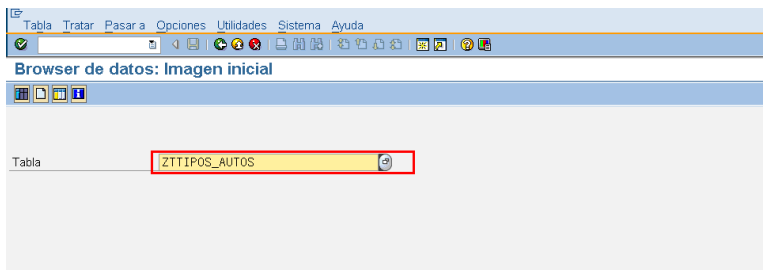
Grabación en memoria intermedia	
<input checked="" type="radio"/>	Imposible grabar en MI
<input type="radio"/>	Permitido grabar en MI, desactivado
<input type="radio"/>	Grabación en MI activ.

Vamos al menú Detalles/Categoría de ampliación y seleccionamos Ampliable de cualquier manera. Grabamos y activamos.



Ejercicio N°44 - Llenar una tabla de datos (SE16).

Para el llenado de las tablas se utiliza la transacción SE16, buscamos el nombre de la tabla que habíamos creado en el ejemplo anterior y pulsamos Enter.



CAPITULO 6 OPEN SQL - Consultas ABAP

Introducción a OPEN SQL.

Es conocido en ABAP **OPEN SQL** a las sentencias SQL "Structured Query Language" que son utilizadas para trabajar con tablas de base de datos. En este capítulo veremos los 10 ejemplos más utilizados.

Las sentencias con:

- SELECT (Selección de datos)
- INSERT (Insertamos datos)
- UPDATE (Actualizamos datos)
- OPEN CURSOR (Abre un cursor)
- FETCH (Avanza una posición del cursor)
- CLOSE CURSOR (Cierra el cursor)
- COMMIT WORK (Actualiza los datos en la tabla)
- ROLLBACK WORK (Deshacer cambios en la tabla)

Ejercicio N°45 - Consulta básica directa.

Este ejemplo muestra una consulta básica donde le seleccionaremos todos los campos de la tabla utilizaremos una tabla interna que le haremos un condición IF para practicar dos conceptos.

REPORT ZSQL01.

* Creamos la estructura de la tabla MARAV

TABLES: MARAV.

DATA: BEGIN OF ST_MARAV.

INCLUDE STRUCTURE MARAV.

DATA END OF ST_MARAV.

Ejercicio N°46 - Consulta básica con TI.

Este ejemplo muestra una consulta básica donde le seleccionaremos todos los campos de la tabla utilizaremos una tabla interna que le haremos un condición IF para practicar dos conceptos.

REPORT ZSQL01.

* Creamos la estructura de la tabla MARAV

TABLES: MARAV.

DATA: BEGIN OF ST_MARAV.

INCLUDE STRUCTURE MARAV.

DATA END OF ST_MARAV.

* Creamos la tabla interna con cabecera

DATA INT_MARAV LIKE TABLE OF ST_MARAV WITH HEADER LINE.

* OPEN SQL BASICO

SELECT * FROM MARAV INTO TABLE INT_MARAV.

* Imprimimos los Datos

LOOP AT INT_MARAV.

IF INT_MARAV-NTGEW <> 0.

WRITE: / INT_MARAV-MANDT,sy-vline,

INT_MARAV-MAKTX,sy-vline,

INT_MARAV-NTGEW,sy-vline.

ENDIF.

ENDLOOP.

WRITE: /'Columna 1', sy-vline,

'Columna 2', sy-vline.

i = i + 1.

Ejercicio N°47 - Consulta primeras 100 filas.

Para este ejemplo utilizaremos el ejemplo anterior y solo modificaremos las siguientes líneas. Se mostrará solo las 100 primeras filas.

* OPEN SQL que muestra las primeras 100 Filas

SELECT * FROM MARAV INTO TABLE INT_MARAV UP TO 100 ROWS.

Ejercicio N°48 - Consulta condición.

Para este ejemplo utilizaremos el ejemplo anterior y solo modificaremos las siguientes líneas.

* OPEN SQL solo copia los que cumplen la condición Sector 00.

```
SELECT * FROM MARAV INTO TABLE INT_MARAV  
WHERE SPART = '00'.
```

Ejercicio N°49 - Mostrar primera fila en una tabla

Seleccionaremos la primera fila de la tabla que cumpla la condición siguiente.

```
REPORT ZSQL1.  
TABLES: MARAV.  
SELECT SINGLE * FROM MARAV WHERE MATNR EQ '000000000000000023'.  
IF SY-SUBRC = 0.  
    WRITE: MARAV-MATNR.  
ELSE.  
    WRITE: / 'ERROR'.  
ENDIF.
```

Ejercicio N°50 - Seleccionamos todos .

En este ejemplo no usaremos tablas internas sino que en con el SELECT iremos imprimiendo los valores.

```
REPORT ZSQL1.  
TABLES: MARAV.  
SELECT * FROM MARAV.  
    WRITE: / MARAV-MATNR.  
ENDSELECT.  
IF SY-SUBRC NE 0.  
    WRITE: / 'ERROR'.  
ENDIF.
```

Ejercicio N°51 - Seleccionamos un solo registro

En este ejemplo no usaremos tablas internas sino que en con el SELECT iremos imprimiendo los valores.

```
REPORT ZSQL1.  
TABLES: MARAV.  
SELECT * FROM MARAV.  
    WRITE: / MARAV-MATNR.  
ENDSELECT.  
IF SY-SUBRC NE 0.
```

```
WRITE: / 'ERROR'.  
ENDIF.
```

Ejercicio N°52 - Máximo, Mínimo, Cantidad.

Si tenemos una tabla y queremos saber la cantidad de registros usamos **COUNT**.

```
REPORT ZSQL1.  
TABLES: MARAV.  
DATA CONTADOR TYPE I.  
SELECT COUNT(*) FROM MARAV INTO CONTADOR.  
WRITE: / 'CANTIDAD:',CONTADOR.
```

Para saber el máximo valor de una columna usamos **MAX**

```
REPORT ZSQL1.  
TABLES: MARAV.  
DATA: MAXIMO LIKE MARAV-BRGEW.  
SELECT MAX( BRGEW )  
FROM MARAV INTO (MAXIMO).  
WRITE: / 'MAX PESO:', MAXIMO, 'Kg'.
```

Para saber el menor valor de una columna usamos **MIN**

```
REPORT ZSQL1.  
TABLES: MARAV.  
DATA: MAXIMO LIKE MARAV-BRGEW.  
SELECT MIN( BRGEW )  
FROM MARAV INTO (MAXIMO).  
WRITE: / 'MINIMO PESO:', MAXIMO, 'Kg'.
```

Ejercicio N°53 - SUMA Y PROMEDIO.

Si tenemos una tabla y queremos sumar toda la columna usamos **SUM**.

```
REPORT ZSQL1.  
TABLES: MARAV.  
DATA: SUMA TYPE F.  
SELECT SUM( BRGEW ) FROM MARAV INTO (SUMA).  
WRITE: / 'LA SUMA ES:',SUMA.
```

Para saber el Promedio utilizamos **AVG**

```
REPORT ZSQL1.
TABLES: MARAV.
      DATA: PROMEDIO TYPE F.
      SELECT AVG( BRGEW ) FROM MARAV INTO
(PROMEDIO) .
WRITE: / 'EL PROMEDIO:', PROMEDIO.
```

Ejercicio N°54 - Buscando cadenas que contengan otras.

Para buscar los responsables de materiales que comienzan con la letra M lo hacemos así.

```
REPORT ZSQL1.
      TABLES: MARAV.
      SELECT * FROM MARAV
      WHERE AENAM LIKE 'M%'.
      WRITE: / MARAV-AENAM.
ENDSELECT.
```

Ejercicio N°55 - Buscando cadenas con listas. IN

Para buscar los materiales que cumplan condiciones que pueden ser listas usamos la sentencia IN es como usar el AND pero permite configurar varios valores.

```
REPORT ZSQL1.
TABLES: MARAV.
      SELECT * FROM MARAV
      WHERE BRGEW
      IN (10,100,1000,2000) .
      WRITE: / MARAV-BRGEW.
ENDSELECT.
```

Ejercicio N°56 - Seleccionando un rango BETWEEN.

Si tenemos que en un campo numérico buscar los que se encuentren entre dos valores los hacemos así. En este ejemplo los materiales que tengan peso en 100 y 1000.

```
REPORT ZSQL1.
TABLES: MARAV.
SELECT * FROM MARAV
WHERE BRGEW BETWEEN 100 AND 1000.
WRITE: / MARAV-BRGEW.
ENDSELECT.
```

Ejercicio N°57- Mostrar tabla ordenada.

Si queremos una campo ordenando utilizamos la sentencia **ORDER BY**.

```
REPORT ZSQL1.
TABLES: MARAV.
SELECT * FROM MARAV ORDER BY AENAM.
WRITE:/ MARAV-AENAM.
ENDSELECT.
```

Ejercicio N°58 - Mostrar todos los iconos en SAP

Si queremos implementar en nuestras pantallas de selección iconos podemos tener un listado completo de iconos.

```
REPORT ZICO.
TABLES: ICON.
SELECT * FROM ICON.
WRITE :/
ICON-name,
33 '@',
34 ICON-id+1(2),
33 '36',
40 ICON-id.
ENDSELECT.
```

Ejercicio N°59- Consulta a dos tablas en SAP.

Si queremos hacer una consulta que me muestre campos de dos tablas en SAP utilizamos. Usaremos la tabla **TRDIR** que contiene todo

los programas y la tabla **TSTC** que contiene todas las transacciones en **BY**.

```
REPORT ZTRANSA.
* Declaración de la estructura.
TYPES: BEGIN OF T_TRAB,
        NAME LIKE TRDIR-NAME,      "CAMPO1 DE TRDIR
        SUBC LIKE TRDIR-SUBC,      "CAMPO1 DE TRDIR
        PGMNA LIKE TSTC-PGMNA,    "CAMPO1 DE TSTC
        TCODE LIKE TSTC-TCODE,    "CAMPO2 DE TSTC
END OF T_TRAB.
* Declaración de la tabla interna
DATA: IT_TRAB TYPE TABLE OF T_TRAB WITH HEADER
LINE.
SELECT TRDIR~NAME
        TRDIR~SUBC
        TSTC~PGMNA
        TSTC~TCODE
        INTO TABLE IT_TRAB
        FROM TRDIR INNER JOIN TSTC ON ( TRDIR~NAME =
TSTC~PGMNA )
        WHERE
                TRDIR~NAME LIKE 'Y%'
        OR
                TRDIR~NAME LIKE 'Z%'
        AND
                TRDIR~SUBC EQ '1'.
LOOP AT IT_TRAB.
        WRITE : / IT_TRAB-NAME, IT_TRAB-TCODE.
ENDLOOP.
```

CAPITULO 7 Ampliaciones y Notas

Ejercicio N°60 - Hola Mundo en ABAP/4.

En este nuestro primer programa te enseñare como crear un desarrollo z, solamente utilizaremos las función de

CAPITULO 8 Programación Orientado a Objetos.

¿Qué es la programación orientado a objetos ?

La programación orientado a objetos **POO** es un paradigma de programación que usa clases y objetos describiendo de forma más real las aplicaciones.

CLASE: orientado a objetos **POO** es un paradigma de programación

Ejercicio N°61 - Hola Mundo de POO.

En este ejemplo haremos un reporte orientada a objeto en **ABAP**, con un método público que imprimirá un mensaje en pantalla.

```
REPORT ZPOO.
* Definimos la clase
CLASS C_MICLASE DEFINITION.
    PUBLIC SECTION.
        METHODS: IMPRIMIR.
    PRIVATE SECTION.
        DATA: CONTADOR TYPE I VALUE 123.
ENDCLASS.
* Hacemos la implementación
CLASS C_MICLASE IMPLEMENTATION.
    METHOD IMPRIMIR.
        CONTADOR = CONTADOR + 1.
        WRITE :/ 'El valor es:' , CONTADOR.
    ENDMETHOD.
ENDCLASS.
* Instanciamos la clase
* Bloque que se ejecutará tras pantalla de selección
START-OF-SELECTION.
    DATA: MYCLASE TYPE REF TO C_MICLASE.
    CREATE OBJECT MYCLASE.
    CALL METHOD MYCLASE->IMPRIMIR.
END-OF-SELECTION.
```


Ejercicio N°62 - POO con métodos y parámetros

En este ejemplo haremos un reporte con métodos y parámetros por los cuales compartiremos datos.

```
REPORT ZPOOL.
* Definicion de clase
CLASS C_SUMA DEFINITION.
  PUBLIC SECTION.
    CLASS-METHODS: class_constructor.
    METHODS: SUMAR IMPORTING VALUE (VALOR) TYPE I,
              RESTAR_DOS,
              ESTADO,
              RESULTADO EXPORTING VALUE (VALOR) TYPE I.
  PRIVATE SECTION.
    DATA: VAR1 TYPE I VALUE 10,
           VAR2 TYPE I VALUE 10.
ENDCLASS.
* Implementamos el método de la clase
CLASS C_SUMA IMPLEMENTATION.
  METHOD class_constructor.
    WRITE: / 'Esto es un ejemplo de clases'.
  ENDMETHOD.

  METHOD SUMAR.
    VAR1 = VAR1 + VALOR.
    VAR2 = VAR2 + VALOR.
  ENDMETHOD.

  METHOD ESTADO.
    WRITE: / VAR1, VAR2.
  ENDMETHOD.

  METHOD RESTAR_DOS.
    VAR1 = VAR1 - 2.
  ENDMETHOD.

  METHOD RESULTADO.
    VALOR = VAR1 + VAR2.
    WRITE: / VALOR.
  ENDMETHOD.
ENDCLASS.
* Creamos la instancia de la clase
START-OF-SELECTION.
  DATA: MYCLASE TYPE REF TO C_SUMA.
  CREATE OBJECT MYCLASE.
```

```

CALL METHOD MYCLASE->ESTADO.
CALL METHOD MYCLASE->RESTAR_DOS.
CALL METHOD MYCLASE->ESTADO.
CALL METHOD MYCLASE->SUMAR( 12 ).
CALL METHOD MYCLASE->ESTADO.
CALL METHOD MYCLASE->RESULTADO.
END-OF-SELECTION.

```

Ejercicio N°63 - POO Herencia de clases.

En este ejemplo veremos cómo hacer herencia de clases en ABAP.

```

REPORT ZP002.
CLASS PAPA DEFINITION.
PUBLIC SECTION.
METHODS: ESTADO.
PRIVATE SECTION.
DATA: VAR1 TYPE I VALUE 10.
ENDCLASS.
* Implementamos el método de la clase
CLASS PAPA IMPLEMENTATION.
METHOD ESTADO.
VAR1 = VAR1 + 2.
WRITE: / VAR1.
ENDMETHOD.
ENDCLASS.
* Creamos la clase HIJO que hereda de PAPA
CLASS HIJO DEFINITION INHERITING FROM PAPA.
ENDCLASS.
* Creamos la instancia de la clase
START-OF-SELECTION.
DATA: MYCLASE TYPE REF TO HIJO.
CREATE OBJECT MYCLASE.
CALL METHOD MYCLASE->ESTADO.
CALL METHOD MYCLASE->ESTADO.
END-OF-SELECTION.

```


CAPITULO 8 Programación de diálogos

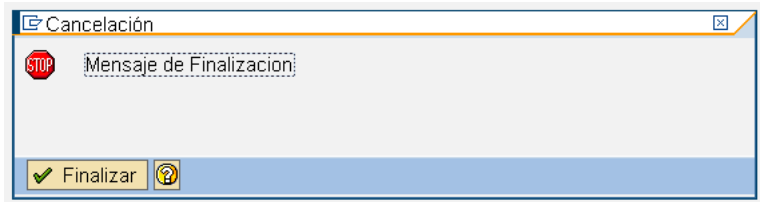
Ejercicio N°64- Mensajes en ABAP

Los mensajes en ABAP/4 se utilizan para informar al usuario de algún dato relevante que tiene que saber existen diferentes tipos de mensajes.

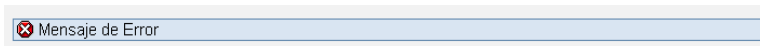
Tabla de tipos de mensajes en ABAP.

T	Descripción	Longitud por defecto
A	Cancelación	El mensaje aparece en un cuadro de dialogo y el programa termina.
E	Error	El mensaje aparece en un cuadro de dialogo y el programa continua
I	Información	Se muestra un cuadro de dialogo de información.
S	Estado	El mensaje se muestra en la barra de estado el programa continua.
W	Advertencia	Depende del contexto un mensaje de error aparece y el programa puede terminar
X	Salir	No muestra ningún mensaje y el programa termina.

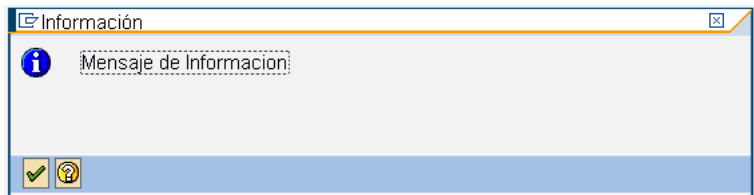
```
REPORT  Z_MENSAJEA_A.  
MESSAGE 'Mensaje de Finalizacion' TYPE 'A'.
```



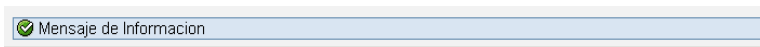
```
REPORT Z_MENSAJEA_E.  
MESSAGE 'Mensaje de Error' TYPE 'E'.
```



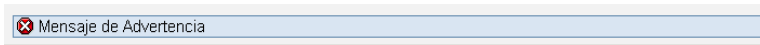
```
REPORT Z_MENSAJEA_I.  
MESSAGE 'Mensaje de Informacion' TYPE 'I'.
```



```
REPORT Z_MENSAJEA_S.  
MESSAGE 'Mensaje de Informacion' TYPE 'S'.
```

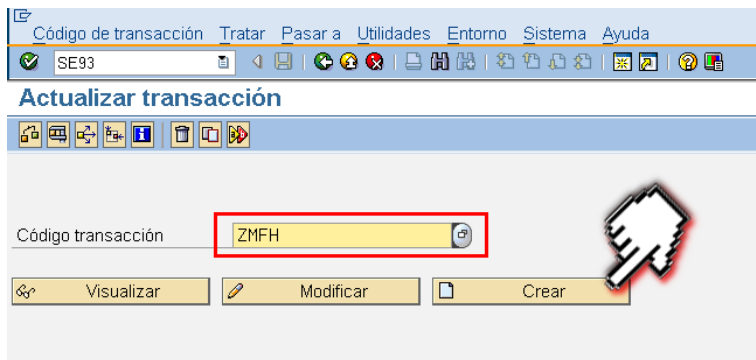


```
REPORT Z_MENSAJEA_S.  
MESSAGE 'Mensaje de Advertencia' TYPE 'W'.
```



Ejercicio N°65 - Creando una transacción en SAP (SE93)

Para crear una transacción se utiliza la transacción **SE93**, le colocamos el código de la transacción que queremos crear, este código siempre tiene que empezar por "Z" ejemplo en este caso crearemos la "ZMFH". En el campo de texto Código de la transacción colocamos nuestro código y pulsamos el botón crear.



En la próxima ventana nos pedirá que le pongamos un texto breve que describa la transacción.

Crear transacción

Código de transacción: ZMFH

Atributos transacción

Texto breve: Ejemplo de una Transaccion

Objeto inicio

- ☒ Programa y dynpro (transacción de diálogo)
- ☐ Programa e imagen de selección (transacción de report)
- ☐ Método de una clase(transacción 00)
- ☐ Transacción con variantes (transacción de variante)
- ☐ Transacción con parámetros (transacción de parámetros)

OK Cancel

En la próxima llenar los siguientes datos y pulsar guardar.

Código transacción Tratar Pasar a Utilidades Entorno Sistema Ayuda

Crear Transacción di...

Código de transacción

Paquete

Texto transacción

Programa

Nº dynpro

Objeto autorización

☒ Se permite actualizar la variante de transacción estándar.

Clasificación

Clasificación de transacción

☒ Transacc.usuario profesional

☐ Transacción EasyWeb

☐ Activo globalm.

Capacidad GUI

☒ SAP GUI para HTML

☒ SAP GUI para Java

☒ SAP GUI para Windows

En la próxima ventana seleccionamos objeto local. Una vez realizado este cambio hemos terminado de crear la transacción **ZMFH** si lo quieres probar entra la transacción y comprueba como abre el programa enlazado.

Sistema origen

Idioma m Español

Ejercicio N°66 - Colores en un reporte Z.

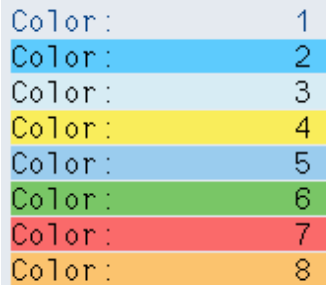
En determinadas ocasiones es necesario diferencial con colores las líneas de un reporte.

```
REPORT ZCOLOR1.  
  FORMAT INTENSIFIED COLOR = 5.  
WRITE: 'Color Verde:'.
```

Ejercicio N°67 - Varios colores en un reporte Z.

En determinadas ocasiones es necesario diferencial con colores las líneas de un reporte.

```
REPORT ZCOLOR2.  
DATA: MCOLOR TYPE I.  
DO 8 TIMES.  
  FORMAT INTENSIFIED COLOR = MCOLOR.  
  MCOLOR = MCOLOR + 1.  
  WRITE: /'Color:',MCOLOR.  
ENDDO.
```



Color:	1
Color:	2
Color:	3
Color:	4
Color:	5
Color:	6
Color:	7
Color:	8

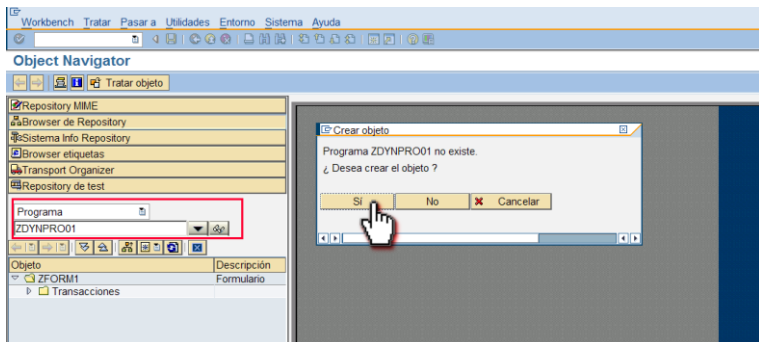
CAPITULO 9 DYNPRO

Introducción a una DYNPRO

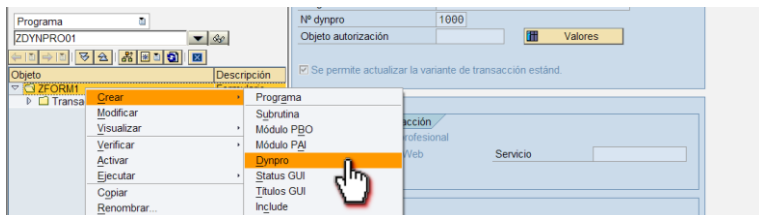
Una **DYNPRO** es un recursos tecnológico de ABAP/4 para crear la capa de presentación de un programa.

Ejercicio N°68- Creación de una DYNPRO (SE80)

Para crear una **DYNPRO** iremos a la transacción **SE80** que es el **Object Navigator** mediante la barra de comandos. Seleccionamos Programa y colocamos el nombre ZDYNPRO, pulsamos Enter y en la próxima ventana nos preguntará que si deseamos crear el programa pulsamos SI y lo guardamos.



Vamos al menú y pulsando click derecho vamos a crear y seleccionamos Dynpro. Nos preguntará por el numero ponemos 0100.



En la siguiente venta nos preguntará por una descripción breve de la Dynpro le colocaremos "Mi nueva dynpro" y seleccionamos en tipo de Dynpro normal. Cuando llenemos estos datos la guardamos en el arbol de la izquierda podrás apreciar que aparece una carpeta llamada Dynpros y dentro de ella el número de tu Dynpro.

The screenshot shows a software configuration window titled 'Nº dynpro' with a value of '100' and a label 'nuevo(revisado)'. It has three tabs: 'Atributos', 'Lista elem.', and 'Lóg.proceso'. The 'Atributos' tab is active. It contains several fields: 'Descripción breve' with the value 'Mi nueva dynpro', 'Idioma maestro' with 'ES' and 'Español', 'Última modif.' and 'Última generación' both with '00:00:00'. Below these is a section 'Tipo de dynpro' with radio buttons for 'Normal' (selected), 'Subscreen', 'Ventana diálogo modal', and 'Dynpro selección'. To the right is an 'Opciones' section with checkboxes for 'Retener datos', 'Desactivar compr.tmpo.ejec.', 'Modelo: No ejecutable', 'Mantener posición desplazam.', and 'S/barr.herram.aplicación'.

Tipos de Dynpros:

Normal: Es la dynpro estandar

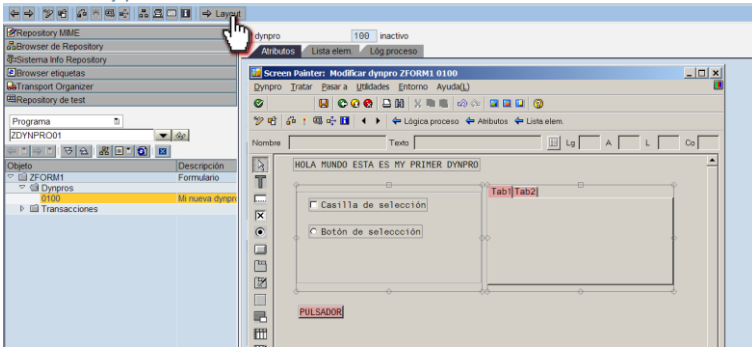
Subscreen: Cuando es una ventana que se abre desde otra Dynpro.

Ventana dialogo modal: Es una pantalla que se utiliza para mostrar mensajes.

Dynpro selección: Son los que se declaran cuando hacemos un selection-screen.

Cuando estemos en el **Object Navigator** pulsamos en el boton superior que dice Layout y se nos abra el **Screen Painter** que lo utilizaremos para adicionar componentes a nuestra Dynpro, tomamos los componentes de la izquierda y lo arrastramos colocándole un nombre y una descripción, por último guardamos y ejecutamos nuestro programa que mostrará todos los elementos visuales que arrastramos anteriormente en el orden colocado.

Screen Painter: Dynpro de ZFORM1 Modificar



CAPITULO 10 Batch Inputs

Introducción a una Batch Inputs SM35

Una **Batch Inputs** es un recursos tecnológico que permite grabar los pasos de una transacción para poder entrarle valores de forma masiva para hacer una Batch Inputs tenemos que ir la transacción SM35 y ponemos a grabar marcando una transacción como objetivo, una vez que finalice te mostrara un proceso detallado de todos los pasos que puedes ejecutar.

CAPITULO 11 Formularios.

Tipos de formularios en SAP

En SAP existen tres tipos de formas de hacer formularios ellas son:

SapScript: Es la forma más antigua de hacer formulario se utiliza desde las primeras versiones de SAP. Se accede a ella desde la transacción SE71.

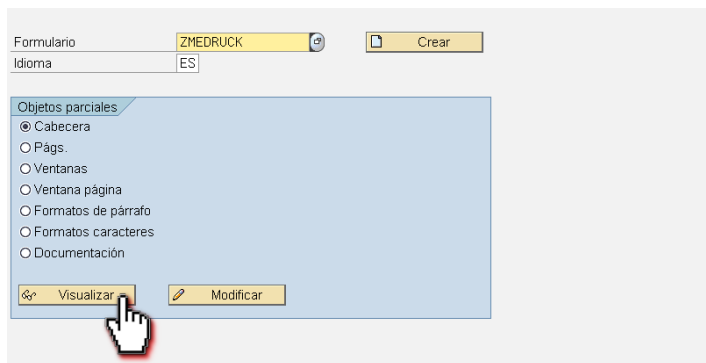
SAPScript es un lenguaje que se utiliza para diversas funciones en SAP entre ellas crear formularios.

SmartForms: Es mucho más fácil de trabajar que los anteriores, está disponible desde la versión 4.7 de SAP.

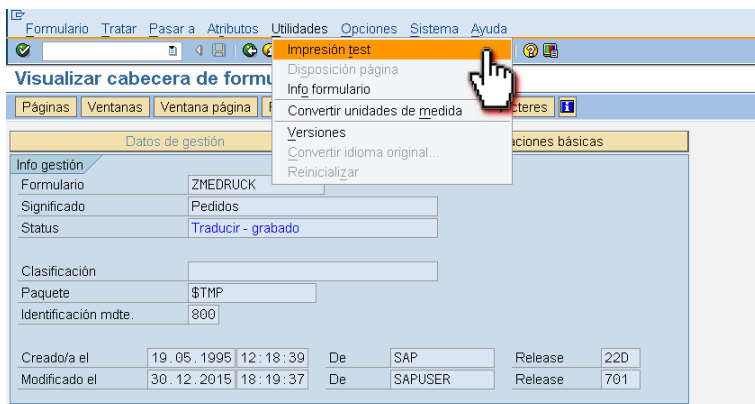
AdobeForms: Es lo último en tecnología de formulario de SAP, surge por una alianza estratégica entre la empresa Adobe y SAP.

Ejercicio N°69 - Mostrar un formulario SAPScript (SE71).

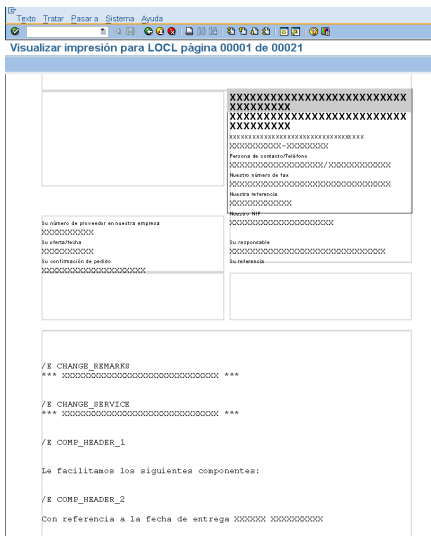
Para mostrar un formulario iremos a la transacción **SE71** o entramos a la ruta: Menú SAP / Herramientas / Imprimir Formularios / SAPscript / SE71 Formularios. Colocamos ZMEDRUCK y pulsamos el botón Visualizar.



En el Menú [Utilidades][Impresión test] hacemos una prueba de impresión del formulario.

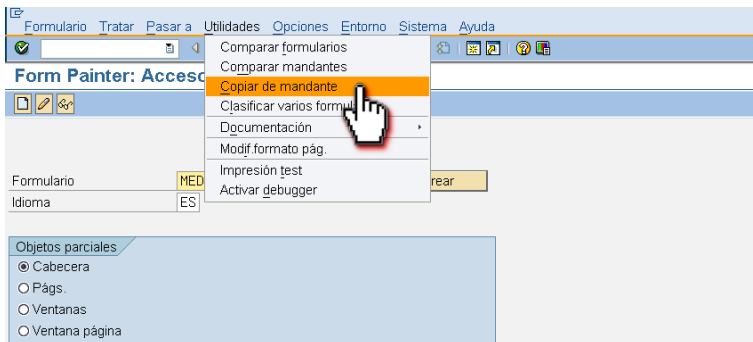


Una vez que definimos la impresora LOCL (Impresora local) podemos ver una vista previa del configuración del formulario.

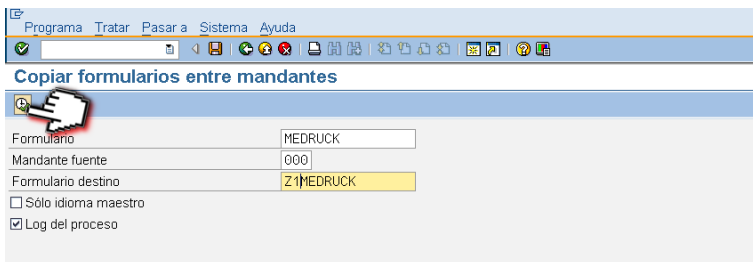


Ejercicio N°70 - Copiar un Formulario SAPScript.

Para copiar un formulario iremos a la transacción **SE71** en Formularios. Colocamos MEDRUCK en el campo formulario. En el Menú superior seleccionamos [**Utilidades**] [**Copiar de mandante**]



En formulario colocamos el nombre del formulario que queremos copiar y en el destino el nuevo nombre en este caso le colocaré: **Z1MEDRUCK**.



Si todo salió bien saldrá el siguiente mensaje .

Copiar formularios entre mandantes

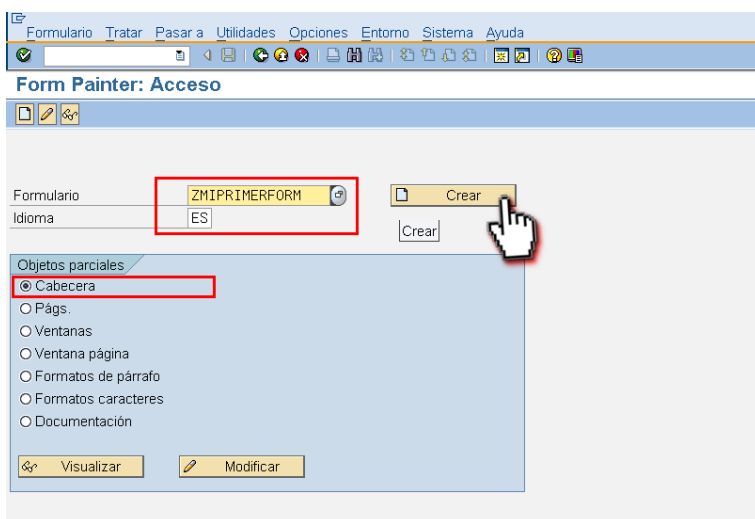
Copiar formularios entre mandantes

```
Z1MEDRUCK : El idioma original ha sido fijado en D.  
Z1MEDRUCK : La definición D ha sido copiada.  
Z1MEDRUCK : El idioma K ha sido copiado.  
Z1MEDRUCK : El idioma L ha sido copiado.
```


Ejercicio N°71 - Crear un Formulario con SAPScript.

Para crear nuestro primer formulario iremos a la transacción **SE71** o entramos a la ruta: Menú SAP / Herramientas / Imprimir Formularios / SAPscript / SE71 Formularios.

Cuando estemos en la transacción seleccionamos en Objetos parciales la opción de cabecera. Colocamos el nombre del formulario "**ZMIPRIMERFORM**", definimos el idioma "**ES**" y pulsamos el botón crear.



Los Objetos parciales que aparecen abajo son:

Cabecera: Se define los datos de configuración del formulario como son fuente, forma, tamaño.

Págs: Se define las páginas que tendrá el formulario.

Ventanas: Se define las ventanas de cada página colocaremos la información que mostraremos dentro de las páginas.

Formatos de párrafos: Se define el formato que tendrá el párrafo del reporte.

En la próxima ventana nos informa que el formulario no existen y aceptamos.

En el significado colocamos una descripción del formulario lo guardamos como objeto local.

Formulario Tratar Pasara Atributos Utilidades Opciones Sistema Ayuda

Modificar cabecera de formulario: ZMIPRIMERFORM

Páginas Ventanas Ventana página Formatos de párrafo Formatos de caracteres

Info gestión

Formulario ZMIPRIMERFORM

Significado Mi primer formulario SAPScript

Status Nuevo - sin grabar

Clasificación

Paquete

Identificación mdte. 800

Creado/a el 00:00:00 De Release

Modificado el 00:00:00 De Release

Atributos idioma

Clave de idioma ES

Idioma original ES

Traducir

☒ A todos los idiomas

☐ A un idioma

☐ No traducir

Los siguiente es crear las paginas vamos al botón Páginas o pulsamos **F6**.

Ahora crearemos la primera página del formulario en el menu superior [**Tratar**] [**Crear Elemento**], en atributos des estándar colocar Páginas "PAGINA1" y en Significado "PAGINA PRINCIPAL DEL FORM" y guardamos los cambios en el botón de guardar superior.

Formulario Tratar Pasar a Atributos Utilidades Opciones Sistema Ayuda

Modificar páginas de formulario: ZSAPMARLON

Ventanas Ventana página Formatos de párrafo Formatos de caracteres

Páginas

Pág.	Significado	Pág.subs	Modo	Tp.num.
PAGINA1	PAGINA PRINCIPAL DEL FORM		INC	ARABIC

Pág. 1 de 1

Atributos estándar

Página: PAGINA1 Significado: PAGINA PRINCIPAL DEL FORM

Página subsig:

Atributos impresión

Nombre recurso:

Modo impresión: ☐

Contador de páginas

Modo: INC

Tipo numeración: ARABIC

Longitud salida:

☐ Mayúsculas

Ahora iremos al botón superior y seleccionamos Ventana o pulsamos la tecla F7. La ventana la usaremos para mostrar los campo que necesitemos como puede ser, Título, Fecha Actual y otras. Para crear la venta en la parte inferior donde dice Atributos estándar hay que adicionar estos campos **MAIN** y en la descripción ponemos tal como se muestra en la siguiente imagen **Ventana pral**. Después de realizar estos cambios guardamos el formulario.

Formulario Tratar Pasar a Atributos Utilidades Opciones Sistema Ayuda

Modificar ventanas de páginas de formulario: ZMIPRIMERFORM

Páginas Ventanas Formatos de párrafo Formatos de caracteres

Ventana página

Página: PAGINA1

Ventana	Significado	Izq.	Arr.	Ancho	Alt.
MAIN	00 Ventana pral.	1,00 CM	1,00 CM	20,00 CM	20,00 CM

Ventana pág. 1 de 1

Atributos estándar

Ventana: MAIN Significado: Ventana pral.

Tipo ventana: MAIN ☐

Margen izqdo.: 1,00 CM Ancho ventana: 20,00 CM

Margen superior: 1,00 CM Altura ventana: 20,00 CM

El siguiente paso es crear los **FORMATOS DE PARRAFOS** donde definiremos el formato de párrafo que utilizaremos en el reporte. Para crear un párrafo vamos al menú superior [**Tratar**] / [**Crear Elemento**] o pulsamos **Shift + F6**. Le ponemos en Formato de párrafo T1 y en Significado: Párrafo de muestra.

Vamos a parametrizaciones básicas y llenamos Pagina inicial y el párrafo por defecto.

Lo último que hacemos es guardarlo y activarlo en menú superior [**Formulario**] [**Activar**], si queremos visualizar el formulario tenemos que ir al menú superior [**Utilidades**] [**Impresión test**] y

comprobamos que no tenga errores tiene que salir el formulario en blanco.

Ejercicio N°59 - Ejecutar un Form SAPScript.

Los Formularios se abren a partir de un programa ABAP/4 y se utiliza la función "OPEN_FORM" para este ejemplo utilizaremos los datos del ejercicio anterior donde creamos un formulario en blanco.

Datos:

FORM: ZMIPRIMERFORM

PAGE: PAGINA1

VENTANA: MAIN

PARRAFO: T1

ELEMENTO: E1

Creamos nuestro programa Z con la transacción SE38 y le colocamos como nombre del programa ZRUNFORM1.

CAPITULO 13 IDocs (Intermediate Document)

Introducción a IDocs

Un **IDocs** es un archivo de texto plano con registros que permite intercambiar información entre diferentes sistemas, para saber la información de los **IDocs** se puede hacer visitando la tabla EDIDC del diccionario de datos, el número es DOCNUM.

Ejercicio N°74 - Hola Mundo en ABAP/4.

En este nuestro primer programa te enseñare como crear un desarrollo z, solamente utilizaremos las función de

Agradecimientos

Quiero agradecer el apoyo de mi esposa Yury que sin ella sería imposible terminar este libro. A mi amiga Yenny por ayudarme a conocer a Chile siempre le estaré agradecido. A mi papá por enseñarme a estudiar todos los días.

Transacciones más utilizadas en SAP.

- FI -

FS00	Datos maestro Cuenta
XK03	Datos maestro de proveedor
XD03	Datos maestro de clientes
FB60	Factura a proveedor (Acreedor)
FB03	Visualización de registro contable
FBL1N	Cuenta corriente de proveedor
F-58	Pago a proveedor
F-44	Compensar
FB08	Anulación de registro
FB70	Facturar a cliente
FBL5N	Cuenta corriente deudor
F-28	Pago a Deudor
FB75	Nota de crédito
FB50	Contabilización Libro mayor
FB03	Visualización de registro contable
FS10N	Visualización de saldos
OB52	Cierre periodo contable
S_ALR_87012249	Informe libro mayor
F28	Pago a cliente

- CO -

KA03	Clase de costos
KS03	Centro de costo
S_ALR_87013611	Reporte de centro de costos

- MM -

MM03	Datos maestros de materiales
ME21N	Creación de un pedido de compra
MIGO	Entrada de mercancía
MIRO	Facturación Proveedor
- SD -	
VA01	Creación de un pedido de venta
VA03	Ver el pedido
VL01N	Salida de mercancía
VF01	Facturación a Cliente
- BASIS -	
SPRO	Parametrización del sistema
SU01	Administración de usuario
PFCG	Crear permisos para los usuarios
AL08	Mostrar usuarios conectados a mandantes
SM04	Mostrar usuarios conectados
- PM -	
SPRO	Parametrización del sistema

Variables del Sistema

SY-ABCDE	CONSTANT: Alfabeto (A,B,C,...)
SY-APPLI	Aplicaciones SAP
SY-BATCH	Batch activo (X)
SY-BATZD	SUBMIT fondo: Diario
SY-BATZM	SUBMIT fondo: Mensual
SY-BATZO	SUBMIT fondo: Unico
SY-BATZS	SUBMIT batch: Inmediatamente
SY-BATZW	SUBMIT fondo: Semanal
SY-BINPT	Batch input activo (X)
SY-BREP4	SUBMIT fondo: Nombre de raíz del report de llamada
SY-BSPLD	SUBMIT fondo: Salida de lista en SPOOL
SY-CALLD	Call modo activo (X)
SY-CALLR	IMPRIMIR: ID para funciones de diálogo
SY-CCURS	Tipo cambio/Campo resultado CURRENCY
CONVERT	
SY-CCURT	Tipo de cambio en tabla de aplicación CURRENCY
CONVERSION	
SY-CDATE	Fecha de tipo de cambio de CURRENCY CONVS.
SY-COLNO	Columna actual en la creación de la lista
SY-CPAGE	Número de página actual
SY-CPROG	RUNTIME: Programa principal
SY-CTABL	Tabla de tipo de cambio en CURRENCY
CONVERSION	
SY-CTYPE	Tipo de cambio 'M','B','G' de CURRENCY
CONVERSION	
SY-CUCOL	Posición del cursor (columna)
SY-CUROW	Posición del cursor (línea)
SY-DATAR	Indicador: Datos recibidos
SY-DATLO	Fecha local, en relación con el usuario
SY-DATUM	SYSTEM: Fecha del día
SY-DATUT	Fecha global, en relación con UTC
SY-DAYST	¿ Horario de verano activo ?

SY-DBCNT Cantidad elementos en conjunto tratado para operaciones BD

SY-DBNAM Base de datos lógica en report ABAP/4

SY-DBSYS SYSTEM: Sistema de base de datos

SY-DCSYS SYSTEM: Sistema de diálogo

SY-DSNAM RUNTIME: Nombre del set de datos para salida en SPOOL

SY-DYNGR Grupo de dynpros del dynpro actual

SY-DYNNR Número de la imagen en pantalla actual

SY-FDAYW Día de semana en el calendario de fábrica

SY-FDPOS Lugar de hallazgo de un string

SY-FFILE INTERNO: Flatfile (USING/GENERATING DATASET)

SY-FLENG Utilización interna (longitud de campo)

SY-FMKEY Menú de códigos de funciones actual

SY-FODEC Utilización interna (campo posiciones decimales)

SY-FOLEN Utilización interna (longitud de salida de campo)

SY-FTYPE Utilización interna (tipo de campo)

SY-GROUP INTERNO: Concatenación

SY-HOST Nombre de la máquina

SY-INDEX Cantidad de repeticiones de bucles

SY-LANGU Clave de idioma para entrar al Sistema SAP

SY-LILLI Número de la línea de lista actual

SY-LINCT Cantidad de líneas de lista

SY-LINNO Línea actual en la creación de una lista

SY-LINSZ Longitud de línea de la lista

SY-LISEL INTERACT.: Línea seleccionada

SY-LISTI Número de la línea de lista actual

SY-LOCDB Existe base de datos local

SY-LOCOP Operación local en base de datos

SY-LOOPC Cantidad de líneas LOOP en steploop de dynpro

SY-LPASS Utilización interna

SY-LSIND Número de la lista de bifurcación

SY-LSTAT INTERACT.: Información de status por nivel de lista

SY-MACOL Cantidad de columnas de instrucción SET MARGIN

SY-MANDT Número de mandante para acceder al Sistema SAP

SY-MARKY	Letra de línea actual para MARK
SY-MAROW	Cantidad de líneas de instrucción SET MARGIN
SY-MODNO	Cantidad de modos alternativos
SY-MSGID	ID de mensaje
SY-MSGLI	INTERACT.: Línea de mensaje (línea 23)
SY-MSGNO	Número del mensaje
SY-MSGTY	Tipo de mensaje (E,I,W,etc.)
SY-MSGV1	Variable en mensaje
SY-MSGV2	Variable en mensaje
SY-MSGV3	Variable en mensaje
SY-MSGV4	Variable en mensaje
SY-NEWPA	Utilización interna
SY-NRPAG	Utilización interna
SY-ONCOM	INTERNO: On Commit Flag
SY-OPSYS	SYSTEM: Sistema operativo
SY-PAART	IMPRESION: Edición
SY-PAGCT	Límite de página de lista en instrucción REPORT
SY-PAGNO	RUNTIME: Página actual en creación de lista
SY-PDEST	IMPRIMIR: Dispositivo de salida
SY-PEXPI	IMPRIMIR: Tiempo de permanencia en SPOOL
SY-PFKEY	RUNTIME: Status de teclas-F actual
SY-PLIST	IMPRESION: Nombre de la orden SPOOL (nombre de lista)
SY-PRABT	IMPRIMIR: Departamento en la portada
SY-PRBIG	IMPRIMIR: Portada de selección
SY-PRCOP	IMPRIMIR: Cantidad de ejemplares
SY-PRDSN	IMPRIMIR: Nombre del set de datos SPOOL
SY-PREFX	Prefijo ABAP/4 para jobs batch
SY-PRIMM	IMPRESION: Salida inmediata
SY-PRNEW	IMPRESION: Nueva orden SPOOL (lista)
SY-PRREC	IMPRIMIR: Destinatario
SY-PRREL	IMPRESION: Borrar tras salida
SY-PRTXT	IMPRIMIR: Texto para portada
SY-REPI2	Utilización interna
SY-REPID	PROGRAM: Nombre de un programa ABAP/4
SY-RSTRT	Utilización interna

SY-RTITL IMPRIMIR: Título de report del programa de impresión
 SY-SAPRL SISTEMA: Release SAP
 SY-SCOLS Columnas en la pantalla
 SY-SLSET Nombre de SELECTON-SETS
 SY-SPONO RUNTIME: Número SPOOL para salida de una lista
 SY-SPONR RUNTIME: Número SPOOL de instrucción
 TRANSFER
 SY-SROWS Líneas en la pantalla
 SY-STACO INTERACT.: Lista visualizada a partir de la columna
 SY-STARO INTERACT.: Lista visualizada a partir de línea
 SY-STEPL Número de la línea LOOP en step dynpro
 SY-SUBCS INTERNO: Status call del report
 SY-SUBRC Valor de retorno tras determinadas sentencias ABAP/4
 SY-SUBTY ABAP: Forma de llamada en SUBMIT
 SY-SYSID SYSTEM: Identificador del Sistema SAP
 SY-TABID Utilización interna
 SY-TABIX RUNTIME: Línea actual de una tabla interna
 SY-TCODE SESSION: Código de transacción actual
 SY-TFDSN RUNTIME: Nombre del set de datos para extractos de
 datos
 SY-TFILL Cantidad actual de entradas en la tabla interna
 SY-TIMLO Hora local, en relación con el usuario
 SY-TIMUT Hora global, en relación con UTC
 SYTITLE PROGRAM: Título del programa ABAP/4
 SY-TLENG Tamaño de la línea de una tabla interna
 SY-TMAXL Cantidad máxima de entradas en la tabla interna
 SY-TNAME Nombre de la tabla interna después de un acceso
 SY-TOCCU Parámetro occurs en tablas internas
 SY-TPAGI Indicador para almacenar tabla interna en bloque paging
 SY-TSTLO Cronomarcador (fecha y hora), en relación con el
 usuario
 SY-TSTUT Cronomarcador (fecha y hora), en relación con UTC
 SY-TTABABC Número de la última línea de tabla interna leída
 SY-TTABI Offset de tablas internas en el área de roll
 SY-TZONE Diferencia de tiempo con 'Hora media de Greenwich'
 (UTC)

SY-UCOMM	INTERACT.: Indicar función en el código OK
SY-ULINE	CONSTANT: Línea de subrayado (_____...)
SY-UNAME	SESSION: Nombre de usuario según entrada a SAP
SY-UZEIT	SYSTEM: Hora
SY-VLINE	CONSTANT: raya vertical
SY-WAERS	T001: Moneda de sociedad tras leer segmento B
SY-WILLI	Número de la línea de ventana actual
SY-WINCO	Posición de cursor en la ventana (columna)
SY-WINDI	Índice de la línea de ventana actual
SY-WINRO	Posición de cursor en la ventana (línea)
SY-WINSL	INTERACT.: Línea en ventana seleccionada
SY-WINX1	Coordenada de ventana (columna izquierda)
SY-WINX2	Coordenada ventana (columna derecha)
SY-WINY1	Coordenada ventana (línea izquierda)
SY-WINY2	Coordenada de ventana (línea derecha)
SY-WTITL	Indicador para cabecera estándar de página
SY-XCODE	Código OK ampliado
SY-ZONLO	Huso horario del usuario

Comando de la barra

- /nend Salir del sistema.
- /nex Salir del sistema y se pierde las entradas que no ha grabado.
- /n Cancelamos la transacción actual.
- /nXXXXX Llamar a otra transacción desde la actual.
- /o Visualiza el resumen de transacciones.
- /i borra la sección actual.

Tecnologías de interfaces utilizadas en SAP.

ALE: Application Link Enabling

BAPI: Business Application Programming Interface

CPI-C: Common Program Interface Communication

EDI: Electronic Data Interchange

HTTP: HyperText Transfer Protocol

LU 6.2: Logical Unit tipo 6.2

RFC: Remote Function Call

OLE: Object Linking and Embedding

SMTP: Simple Mail Transfer Protocol

SOAP: Simple Object Access Protocol

TCP/IP: Transmission Control Protocol / Internet Protocol

XML: Extensible Markup Language

Workbench Abap.

Editor ABAP para tratar el código fuente

Dictionary ABAP para tratar definiciones de tabla de base de datos, tipos de datos centrales, etc.

Screen Painter para configurar pantallas (pantallas junto a funciones para diálogos de usuario)

Menu Painter para diseñar interfaces de usuario (barra de menús, barra de herramientas estándar, barra de herramientas de aplicaciones, parametrizaciones de teclas de función)

Function Builder para actualizar módulos de funciones

Generador de clases para actualizar clases e interfaces globales

Visítenos en la web:

www.marlonfalcon.cl

100

