**Field Symbol**

* Um field symbol é uma referência a uma variável (pode ser uma variável simples, uma estrutura, uma tabela...).
* O field symbol não armazena valor, só faz referência à variável, que por sua vez armazena valor.
* Quando alteramos o valor do field symbol estamos a alterar o valor da variável à qual esse field symbol faz referência e vice-versa.
* O tipo do field symbol tem que ser sempre igual ao da variável.

**Criação de field symbol**

REPORT ZZADRI\_FIELD\_SYMBOL.

DATA ld\_text TYPE char100. (criação de variável para testar o field dymbol)

ld\_text = ‘Teste’.

FIELD-SYMBOLS <ld\_text> TYPE char100. (criação do field symbol. Dentro dos < > põe-se o nome que queremos dar ao field symbol e depois o tipo)

ASSIGN ld\_text TO <ld\_text>. (para associarmos o field symbol à variável)

**Verificar se um field symbol tem referência**

REPORT ZZADRI\_FIELD\_SYMBOL.

DATA ld\_text TYPE char100.

ld\_text = ‘Teste’.

FIELD-SYMBOLS <ld\_text> TYPE char100.

(para verificarmos se tem referência usamos um IF + noe do field + IS ASSIGNED. Neste caso vai dizer que não tem referência porque apaguei a parte do “Assign’)

IF <ld\_text> IS ASSIGNED.

WRITE ‘Tem referência.’.

ELSE

WRITE ‘Não tem referência.’.

ENDIF.

**Limpar uma referência**

REPORT ZZADRI\_FIELD\_SYMBOL.

DATA ld\_text TYPE char100.

ld\_text = ‘Teste’.

FIELD-SYMBOLS <ld\_text> TYPE char100.

ASSIGN ld\_text TO <ld\_text>.

UNASSIGN <ld\_text>. (depois desta instrução, o field symbol fica limpo, sem nenhuma referência)

**Field symbol com variáveis com estruturas**

REPORT ZZADRI\_FIELD\_SYMBOL.

DATA ls\_scarr TYPE scarr.

FIELD-SYMBOLS: <ls\_scarr> TYPE scarr.

ls\_scarr-carrid = 1.

ls\_scarr-carrname = ‘Teste’.

ASSIGN ls\_scarr TO <ls\_scarr>.

(É exatamente igual às variáveis normais. E se mudarmos o valor no field symbol, estamos a a mudar também na estrutura e vice-versa. Tal como nas variáveis normais.)

**Field symbol com variáveis com tabelas**

REPORT ZZADRI\_FIELD\_SYMBOL.

DATA: lt\_scarr TYPE STANDARD TABLE OF scarr.

DATA: ls\_scarr TYPE scarr.

FIELD-SYMBOLS <ls\_scarr> TYPE scarr.

SELECT \*

FROM scarr

INTO TABLE lt\_scarr. (Esta instrução carrega os dados do banco de dados para a tabela interna)

LOOP AT lt\_scarr ASSIGNING <ls\_scarr>. (Este loop vai passar em todas as linhas da tabela interna, uma a uma, e carregar a informação de cada linha para o field symbol)

WRITE <ls\_scarr>-carrname. (neste caso queremos mostrar na tela através do field symbol o nome da companhia)

ENDLOOP.

Desta forma usada em cima, o field symbol está a apontar para cada linha da tabela interna, não está a apontar para a tabela interna diretamente. Para apontar diretamente para a tabela, fazemos assim:

REPORT ZZADRI\_FIELD\_SYMBOL.

DATA: lt\_scarr TYPE STANDARD TABLE OF scarr.

FIELD-SYMBOLS <lt\_scarr> LIKE scarr.

ASSIGN lt\_scarr TO <lt\_scarr>.

SELECT \*

FROM scarr

INTO TABLE lt\_scarr.

**Field symbol com variáveis de outros programas**

REPORT ZZADRI\_FIELD\_SYMBOL.

DATA ld\_text TYPE char100.

DATA ld\_varname TYPE char100.

FIELD-SYMBOLS <ld\_text> TYPE char100.

ld\_text = ‘Teste’

- Imaginemos que esta variável esta noutro programa chamado REPORT ZZADRI\_VARIAVEIS. Para a usarmos para o nosso field symbol, faríamos assim:

- Declarávamos uma variável para guardar o valor da outra variável (criada la em cima - varname)

- Atribuíamos o valor da variável do outro programa à variável que criamos:

ld\_varname = ‘(ZZADRI\_VARIAVEIS)ld\_text’.

- Atribuíamos o valor desta nova variável ao nosso field symbol:

ASSIGN (ld\_varname) TO <ld\_text>.