

FOR ... ENDFOR Command

Executes a set of commands a specified number of times.

Syntax

```
FOR Var = nInitialValue TO nFinalValue [STEP nIncrement]  
Commands  
[EXIT]  
[LOOP]  
ENDFOR | NEXT
```

Arguments

Var

Specifies a variable or an array element that acts as the counter. The variable or array element doesn't have to exist before FOR ... ENDFOR is executed.

nInitialValue TO *nFinalValue*

nInitialValue is the initial value of the counter; *nFinalValue* is the final value of the counter.

STEP *nIncrement*

nIncrement is the amount the counter is incremented or decremented. If *nIncrement* is negative, the counter is decremented. If you omit STEP, the counter is incremented by 1.

Commands

Specifies the Visual FoxPro commands to be executed. *Commands* can include any number of commands.

EXIT

Transfers control from within the FOR ... ENDFOR loop to the command immediately following ENDFOR. You can place EXIT anywhere between FOR and ENDFOR.

LOOP

Returns control directly back to the FOR clause without executing the statements between LOOP and ENDFOR. The counter is incremented or decremented as if ENDFOR were reached. LOOP can be placed anywhere between FOR and ENDFOR.

Remarks

A variable or an array element is used as a counter to specify how many times the Visual FoxPro commands inside the FOR ... ENDFOR loop are executed.

The Visual FoxPro commands after FOR are executed until ENDFOR or NEXT is reached.

The counter *MemVarName* is then incremented by the value of *nIncrement*. If you omit the STEP clause, the counter is incremented by 1.

The counter is then compared with *nFinalValue*. If the counter is less than or equal to *nFinalValue*, the commands following the FOR clause are executed again.

If the counter is greater than *nFinalValue*, the FOR ... ENDFOR loop is exited and program execution continues with the first command following ENDFOR or NEXT.

Note The values of *nInitialValue*, *nFinalValue*, and *nIncrement* are only read initially.

However, changing the value of the counter *MemVarName* inside the loop affects the number of times the loop is executed.

If the value of *nIncrement* is negative and the initial value *nInitialValue* is greater than the final value *nFinalValue*, the counter is decremented each time through the loop.

[Example](#)

FOR ... ENDFOR Command Example

In Example 1, the numbers 1 through 10 are displayed.

Example 2 uses memory variables for the initial, final, and STEP values to display all even-numbered records from 2 through 10 in *customer*.

```
* Example 1
CLEAR
FOR gnCount = 1 TO 10
    ? gnCount
ENDFOR

* Example 2
SET TALK OFF
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && Opens Customer table
STORE 2 TO gnI  && Initial value
STORE 10 TO gnJ  && Final value
STORE 2 TO K  && Step value
FOR gnCount = gnI TO gnJ STEP K
    GOTO gnCount  && Move record pointer
    DISPLAY company && Display company name
ENDFOR
```

DO WHILE ... ENDDO Command

Executes a set of commands within a conditional loop.

Syntax

```
DO WHILE lExpression  
Commands  
[LOOP]  
[EXIT]  
ENDDO
```

Arguments

lExpression

Specifies a logical expression whose value determines whether the commands between DO WHILE and ENDDO are executed.

As long as *lExpression* evaluates to true (.T.), the set of commands are executed.

Commands

Specifies the set of Visual FoxPro commands to be executed as long as *lExpression* evaluates to true (.T.).

LOOP

Returns program control directly back to DO WHILE.

LOOP can be placed anywhere between DO WHILE and ENDDO.

EXIT

Transfers program control from within the DO WHILE loop to the first command following ENDDO.

EXIT can be placed anywhere between DO WHILE and ENDDO.

Remarks

Commands between DO WHILE and ENDDO are executed for as long as the logical expression *lExpression* remains true (.T.).

Each DO WHILE statement must have a corresponding ENDDO statement.

Comments can be placed after DO WHILE and ENDDO on the same line.

The comments are ignored during program compilation and execution.

Example

DO WHILE ... ENDDO Command Example

In the following example, the number of products in stock priced over \$20 is totaled in the DO WHILE loop until the end of the file (EOF) is encountered. The DO WHILE loop is exited and the total is displayed.

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE products  && Opens Products table
SET TALK OFF
gnStockTot = 0

DO WHILE .T.  && Begins loop
    IF EOF( )
        EXIT
    ENDIF
    IF unit_price < 20
        SKIP
        LOOP
    ENDIF
    gnStockTot = gnStockTot + in_stock
    SKIP
ENDDO      && Ends loop

CLEAR
? 'Total items in stock valued over 20 dollars:'
?? gnStockTot
```

SCAN ... ENDSCAN Command

Moves the record pointer through the currently selected table and executes a block of commands for each record that meets the specified conditions.

Syntax

```
SCAN [NOOPTIMIZE]  
[Scope] [FOR lExpression1] [WHILE lExpression2]  
[Commands]  
[LOOP]  
[EXIT]  
ENDSCAN
```

Arguments

NOOPTIMIZE

Prevents Rushmore optimization of SCAN.

For more information, see [SET OPTIMIZE](#) and "Understanding Rushmore Technology" in Chapter 15, [Optimizing Applications](#), in the *Programmer's Guide*.

Scope

Specifies a range of records to be scanned. Only the records within the range are scanned.

The scope clauses are: ALL, NEXT *nRecords*, RECORD *nRecordNumber*, and REST.

بنود نطاق هي: ALL ، NEXT *nRecords* ، RECORD *nRecordNumber* ، و REST .

For more information on scope clauses, see the [Scope Clauses](#) online topic.

The default scope for SCAN is all records (ALL).

FOR *lExpression1*

Executes commands only for records for which *lExpression1* evaluates to true (.T.).

Including the FOR clause lets you filter out records you don't want scanned.

Rushmore optimizes a query created with SCAN ... FOR if *lExpression1* is an optimizable expression .

For best performance, use an optimizable expression in the FOR clause.

For more information, see [SET OPTIMIZE](#) and "Understanding Rushmore Technology" in Chapter 15, [Optimizing Applications](#), in the *Programmer's Guide*.

WHILE *lExpression2*

Specifies a condition whereby the commands are executed for as long as *lExpression2* evaluates to true (.T.).

Commands

Specifies the Visual FoxPro commands to be executed.

LOOP

Returns control directly back to SCAN.

LOOP can be placed anywhere between SCAN and ENDSCAN.

EXIT

Transfers program control from within the SCAN ... ENDSCAN loop to the first command following ENDSCAN.

EXIT can be placed anywhere between SCAN and ENDSCAN.

ENDSCAN

Indicates the end of the SCAN procedure.

Remarks

SCAN automatically advances the record pointer to the next record that meets the specified conditions and executes the block of commands.

You can place comments after ENDSCAN on the same line.

The comments are ignored during program compilation and execution.

[Example](#)

SCAN ... ENDSCAN Command Example

The following example uses a SCAN ... ENDSCAN loop to display all the companies in Sweden.

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && Opens Customer table
CLEAR
```

```
SCAN FOR UPPER(country) = 'SWEDEN'
    ? contact, company, city
ENDSCAN
```