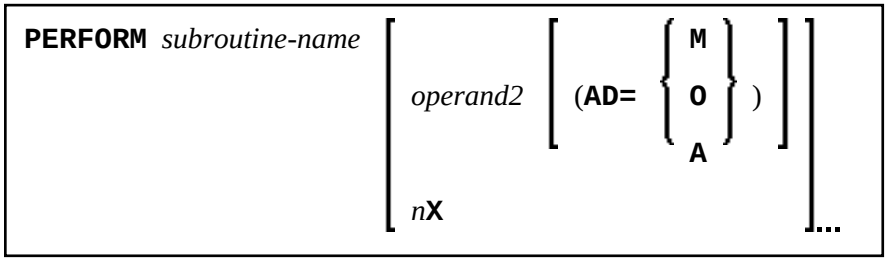
PERFORM



This chapter covers the following topics:

- Function
- Syntax Description
- Examples

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Related Statements: CALL | CALL FILE | CALL LOOP | CALLNAT | DEFINE SUBROUTINE | ESCAPE | FETCH

Belongs to Function Group: *Invoking Programs and Routines*

Function

The PERFORM statement is used to invoke a Natural subroutine.

Nested PERFORM Statements

The invoked subroutine may contain a PERFORM statement to invoke another subroutine (the number of nested levels is limited by the size of the required memory).

A subroutine may invoke itself (recursive subroutine). If database operations are contained within an external subroutine that is invoked recursively, Natural will ensure that the database operations are logically separated.

Parameter Transfer with Dynamic Variables

See the statement CALLNAT.

Syntax Description

Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand2	C	S	A	G		A	U	N	P	I	F	B	D	T	L	C	G	O	yes	yes

Syntax Element Description:

Syntax Element	Description
subroutine-name	**Subroutine to be Invoked:** For a subroutine name (maximum 32 characters), the same naming conventions apply as for user-defined variables. The subroutine name is independent of the name of the module in which the subroutine is defined (it may but need not be the same). The subroutine to be invoked must be defined with a `DEFINE SUBROUTINE` statement. It may be an inline or external subroutine (see `DEFINE SUBROUTINE` statement). Within one object, no more than 50 external subroutines may be referenced. **Data Available in a Subroutine** • **Inline Subroutines** No explicit parameters can be passed from the invoking object to an inline subroutine. An inline subroutine has access to the currently established global data area as well as the local data area defined within the same object module. • **External Subroutines** An external subroutine has access to the currently established global data area. Moreover parameters can be passed with the `PERFORM` statement from the invoking object to the external subroutine (see *operand2*); thus, you may reduce the size of the global data area.

Syntax Element	Description
operand2	**Parameters to be Passed:** When an external subroutine is invoked with the PERFORM statement, one or more parameters (*operand2*) can be passed with the PERFORM statement from the invoking object to the external subroutine. For an inline subroutine, *operand2* cannot be specified. If parameters are passed, the structure of the parameter list must be defined in a DEFINE DATA statement. By default, the parameters are passed "by reference", that is, the data are transferred via address parameters, the parameter values themselves are not moved. However, it is also possible to pass parameters "by value", that is, pass the actual parameter values. To do so, you define these fields in the DEFINE DATA PARAMETER statement of the subroutine with the option BY VALUE or BY VALUE RESULT. ● If parameters are passed "by reference" the following applies: The sequence, format and length of the parameters in the invoking object must match exactly the sequence, format and length of the DEFINE DATA PARAMETER structure of the invoked subroutine. The names of the variables in the invoking object and the subroutine may be different. ● If parameters are passed "by value" the following applies: The sequence of the parameters in the invoking object must match exactly the sequence in the DEFINE DATA PARAMETER structure of the invoked subroutine. Formats and lengths of the variables in the invoking object and the subroutine may be different; however, they have to be data transfer compatible. The names of the variables in the invoking object and the subroutine may be different. If parameter values that have been modified in the subroutine are to be passed back to the invoking object, you have to define these fields with BY VALUE RESULT. With BY VALUE (without RESULT) it is not possible to pass modified parameter values back to the invoking object (regardless of the AD specification; see also below). **Note:** With BY VALUE, an internal copy of the parameter values is created. The subroutine accesses this copy and can modify it, but this will not affect the original parameter values in the invoking object. With BY VALUE RESULT, an internal copy is likewise created; however, after termination of the subroutine, the original parameter values are overwritten by the (modified) values of the copy. For both ways of passing parameters, the following applies: ● In the parameter data area of the invoked subroutine, a redefinition of groups is only permitted within a REDEFINE block. ● If an array is passed, its number of dimensions and occurrences in the subroutine's parameter data area must be same as in the PERFORM parameter list. **Note:** If multiple occurrences of an array that is defined as part of an indexed group are passed with the PERFORM statement, the corresponding fields in the subroutine's parameter data area must not be redefined, as this would lead to the wrong addresses being passed.

Syntax Element	Description
AD=	**Attributes:** If *operand2* is a variable, you can mark it in one of the following ways: AD=O Non-modifiable, see session parameter AD=O. **Note:** Internally, AD=O is processed in the same way as BY VALUE (see Note under *operand2*). AD=M Modifiable, see session parameter AD=M. This is the default setting. AD=A Input only, see session parameter AD=A. If *operand2* is a constant, AD cannot be explicitly specified. For constants, AD=O always applies.
*n*X	**Parameters to be Skipped:** With the notation *n*X you can specify that the next *n* parameters are to be skipped (for example, 1X to skip the next parameter, or 3X to skip the next three parameters); this means that for the next *n* parameters no values are passed to the external subroutine. A parameter that is to be skipped must be defined with the keyword OPTIONAL in the subroutine's DEFINE DATA PARAMETER statement. OPTIONAL means that a value can - but need not - be passed from the invoking object to such a parameter.

Examples

- Example 1 - PERFORM as Inline Subroutine
- Example 2 - PERFORM as External Subroutine

Example 1 - PERFORM as Inline Subroutine

```
** Example 'PEREX1': PERFORM   (as inline subroutine)
************************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 NAME
  2 ADDRESS-LINE  (A20/2)
  2 PHONE
*
1 #ARRAY    (A75/1:4)
1 REDEFINE #ARRAY
  2 #ALINE  (A25/1:4,1:3)
1 #X        (N2) INIT <1>
1 #Y        (N2) INIT <1>
END-DEFINE
*
LIMIT 5
FIND EMPLOY-VIEW WITH CITY = 'BALTIMORE'
  MOVE NAME            TO #ALINE (#X,#Y)
  MOVE ADDRESS-LINE(1) TO #ALINE (#X+1,#Y)
  MOVE ADDRESS-LINE(2) TO #ALINE (#X+2,#Y)
```

```
  MOVE PHONE           TO #ALINE (#X+3,#Y)
  IF  #Y = 3
    RESET INITIAL #Y
    /*
    PERFORM PRINT
    /*
  ELSE
    ADD 1 TO #Y
  END-IF
  AT END OF DATA
    /*
    PERFORM PRINT
    /*
  END-ENDDATA
END-FIND
*
DEFINE SUBROUTINE PRINT
  WRITE NOTITLE (AD=OI) #ARRAY(*)
  RESET #ARRAY(*)
  SKIP 1
END-SUBROUTINE
*
END
```

Output of Program PEREX1:

```
JENSON                   LAWLER                   FORREST
2120 HASSELL             4588 CANDLEBERRY AVE     37 TENNYSON DRIVE
 #206                    BALTIMORE                BALTIMORE
998-5038                 629-0403                 881-3609

ALEXANDER                NEEDHAM
409 SENECA DRIVE         12609 BUILDERS LANE
BALTIMORE                BALTIMORE
345-3690                 641-9789
```

Example 2 - PERFORM as External Subroutine

Program containing PERFORM statement:

```
** Example 'PEREX2': PERFORM (as external subroutine)
************************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 NAME
  2 ADDRESS-LINE  (A20/2)
  2 PHONE
*
1 #ALINE   (A25/1:4,1:3)
1 #X       (N2)            INIT <1>
1 #Y       (N2)            INIT <1>
END-DEFINE
*
LIMIT 5
*
FIND EMPLOY-VIEW WITH CITY = 'BALTIMORE'
  MOVE NAME            TO #ALINE (#X,#Y)
  MOVE ADDRESS-LINE(1) TO #ALINE (#X+1,#Y)
  MOVE ADDRESS-LINE(2) TO #ALINE (#X+2,#Y)
  MOVE PHONE           TO #ALINE (#X+3,#Y)
  IF  #Y = 3
```

```
    RESET INITIAL #Y
    /*
    PERFORM PEREX2E #ALINE(*,*)
    /*
  ELSE
    ADD 1 TO #Y
  END-IF
  AT END OF DATA
    /*
    PERFORM PEREX2E #ALINE(*,*)
    /*
  END-ENDDATA
END-FIND
*
END
```

External subroutine PEREX3 with parameters called by program PEREX2:

```
** Example 'PEREX3': SUBROUTINE (external subroutine with parameters)
************************************************************************
DEFINE DATA
PARAMETER
1 #ALINE  (A25/1:4,1:3)
END-DEFINE
*
DEFINE SUBROUTINE PEREX2E
  WRITE NOTITLE (AD=OI) #ALINE(*,*)
  RESET #ALINE(*,*)
  SKIP 1
END-SUBROUTINE
*
END
```

Output of Program PEREX2:

```
JENSON                    LAWLER                    FORREST
2120 HASSELL              4588 CANDLEBERRY AVE      37 TENNYSON DRIVE
 #206                     BALTIMORE                 BALTIMORE
998-5038                  629-0403                  881-3609

ALEXANDER                 NEEDHAM
409 SENECA DRIVE          12609 BUILDERS LANE
BALTIMORE                 BALTIMORE
345-3690                  641-9789
```