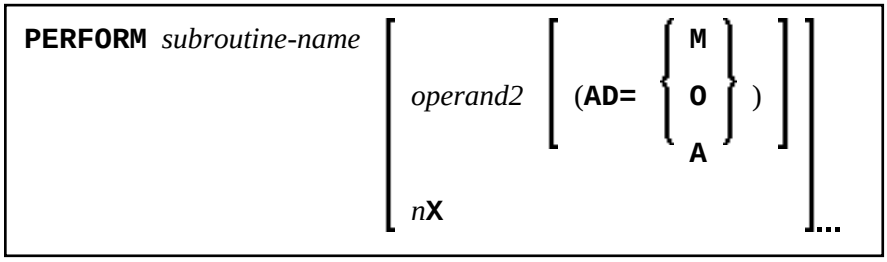
PERFORM



Dieses Kapitel behandelt folgende Themen:

- Funktion
- Syntax-Beschreibung
- Beispiele

Eine Erläuterung der in dem Syntax-Diagramm verwendeten Symbole entnehmen Sie dem Abschnitt *Syntax-Symbole*.

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

Gehört zur Funktionsgruppe: *Aufrufen von Programmen und Unterprogrammen*

Funktion

Das Statement PERFORM dient dazu, eine Natural-Subroutine aufzurufen.

Verschachtelte PERFORM-Statements

Eine aufgerufene Subroutine kann ihrerseits mit einem PERFORM-Statement eine andere Subroutine aufrufen. Wieviele Ebenen eine derartige Verschachtelung mehrerer PERFORM-Statements erreichen darf, hängt vom benötigten Speicherplatz ab.

Eine Subroutine kann auch sich selbst aufrufen (rekursive Subroutine). Falls eine rekursive externe Subroutine Datenbankzugriffe beinhaltet, sorgt Natural automatisch dafür, dass diese als getrennte logische Operationen durchgeführt werden.

Parameter-Übertragung mit dynamischen Variablen

Siehe CALLNAT-Statement.

Syntax-Beschreibung

Operanden-Definitionstabelle:

Operand	Mögliche Struktur					Mögliche Formate												
operand2	C	S	A	G		A	U	N	P	I	F	B	D	T	L	C	G	O

Referenzierung erlaubt	Dynam. Definition
ja	ja

Syntax-Element-Beschreibung:

subroutine-name	**Aufzurufende Subroutine:** Für einen Subroutinen-Namen (maximal 32 Zeichen) gelten dieselben Namenskonventionen wie für Benutzervariablen (siehe *Namenskonventionen für Benutzervariablen* in der Dokumentation *Natural benutzen*). Der Subroutine-Name ist unabhängig vom Namen des Moduls, in dem die Subroutine definiert wird (er kann identisch sein, muss es aber nicht unbedingt sein). Die aufzurufende Subroutine muss mit einem `DEFINE SUBROUTINE`-Statement definiert werden. Es kann sich dabei um eine interne oder externe Subroutine handeln (siehe `DEFINE SUBROUTINE`-Statement). Innerhalb eines Objekts können nicht mehr als 50 externe Subroutinen referenziert werden. **Von einer Subroutine benutzbare Daten:** • **Interne Subroutinen:** Es ist nicht möglich, mit dem `PERFORM`-Statement Parameter explizit vom aufrufenden Programm an eine programmintern definierte Subroutine zu übergeben. Eine programminterne Subroutine kann auf die im selben Objektmodul verwendete Local Data Area sowie auf die derzeit verwendete Global Data Area zugreifen. • **Externe Subroutinen:** Eine programmextern definierte Subroutine kann Daten aus der Global Data Area des aufrufenden Objekts verwenden. Außerdem können Sie mit dem `PERFORM`-Statement Parameter vom aufrufenden Objekt an die aufgerufene Subroutine übergeben (siehe *operand2*); dadurch können Sie die Größe der Global Data Area entsprechend klein halten.

operand2	**Übergabe von Parametern an die externe Subroutine:** Wenn Sie mit dem PERFORM-Statement eine externe Subroutine aufrufen, können Sie mit dem PERFORM-Statement einen oder mehrere Parameter (*operand2*) vom aufrufenden Objekt an die externe Subroutine übergeben. Für interne Subroutinen kann *operand2* nicht angegeben werden. Wenn Parameter übergeben werden, muss die Struktur der Parameterliste in einem DEFINE DATA-Statement definiert werden. Standardmäßig erfolgt die Übergabe der Parameter durch Referenzierung (*By Reference*), d.h. die Daten werden über Adress-Parameter übergeben, die Parameterwerte selbst werden nicht übertragen. Es besteht aber auch die Möglichkeit, Parameter *By Value* zu übergeben, d.h. die Parameterwerte selbst zu übergeben. Hierzu definieren Sie die betreffenden Felder im DEFINE DATA PARAMETER-Statement der Subroutine mit der Option BY VALUE bzw. BY VALUE RESULT). ● Für die Parameterübergabe durch Referenzierung (*By Reference*) gilt: Reihenfolge, Format und Länge der Parameter im aufrufenden Objekt müssen genau den Angaben im DEFINE DATA PARAMETER-Statement der aufgerufenen Subroutine entsprechen. Die Namen der Variablen im aufrufenden Objekt und der Subroutine können unterschiedlich sein. ● Für die Parameterübergabe der Parameterwerte selbst (*By Value*) gilt: Die Reihenfolge der Parameter im aufrufenden Objekt muss der Reihenfolge im DEFINE DATA PARAMETER-Statement der aufgerufenen Subroutine entsprechen. Formate und Längen der Variablen im aufrufenden Objekt und in der Subroutine können unterschiedlich sein, müssen aber datenübertragungskompatibel sein. Die Namen der Variablen im aufrufenden Objekt und in der Subroutine können unterschiedlich sein. Um Parameterwerte, die in der Subroutine verändert wurden, an das aufrufende Objekt zurückgeben zu können, müssen Sie die betreffenden Felder mit BY VALUE RESULT definieren. Mit BY VALUE (ohne RESULT) ist es nicht möglich, veränderte Parameterwerte an das aufrufende Objekt zurückzugeben (unabhängig von der Attribut-Definition (AD-Paramterangabe; vgl. unten). **Anmerkung:** Intern wird bei BY VALUE eine Kopie der Parameterwerte erzeugt. Die Subroutine greift auf diese Kopie zu und kann sie modifizieren, was aber keinen Einfluss auf die Originalparameterwerte im aufrufenden Objekt hat. Bei BY VALUE RESULT wird ebenfalls eine Kopie erzeugt, aber nach Beendigung der Subroutine überschreiben die (modifizierten) Werte der Kopie die Originalparameterwerte. Bei beiden Arten der Parameterübergabe sind folgende Punkte zu beachten: ● Eine Gruppe darf in der Parameter Data Area der aufgerufenen Subroutine innerhalb eines REDEFINE-Statement-Blocks redefiniert werden. ● Bei der Übergabe eines Arrays muss die Anzahl seiner Dimensionen und Ausprägungen in der Parameter Data Area der Subroutine denen in der PERFORM-Parameterliste entsprechen. **Anmerkung:** Wenn mehrere Ausprägungen eines Arrays, das als Teil einer indizierten Gruppe definiert ist, mit dem PERFORM-Statement übergeben werden, dürfen die entsprechenden Felder in der Parameter Data Area der Subroutine nicht redefiniert werden, da sonst die falschen Adressen übergeben werden.

AD=	**Definition von Attributen:** Wenn *operand2* eine Variable ist, können Sie sie folgendermaßen kennzeichnen:	
	AD=O	Nicht modifizierbar, siehe Session-Parameter AD=O. **Anmerkung:** Intern wird AD=O genauso verarbeitet wie BY VALUE (siehe Anmerkung unter *operand2*).
	AD=M	Modifizierbar, siehe Session-Parameter AD=M. Dies ist die Standardeinstellung.
	AD=A	Nur für Eingabe, siehe Session-Parameter AD=A.
	Wenn *operand2* eine Konstante ist, kann AD nicht explizit angegeben werden. Für Konstanten gilt immer AD=O.	
***n*X**	**Angabe zu überspringender Parameter:** Mit der Notation *n*X können Sie angeben, dass die nächsten *n* Parameter übersprungen werden sollen (zum Beispiel 1X, um den nächsten Parameter zu überspringen, oder 3X, um die nächsten 3 Parameter zu überspringen); dies bedeutet, dass für die nächsten *n* Parameter keine Werte an die externe Subroutine übergeben werden. Ein zu überspringender Parameter muss mit dem Schlüsselwort OPTIONAL im DEFINE DATA PARAMETER-Statement der Subroutine definiert werden. OPTIONAL bedeutet, dass ein Wert vom aufrufenden Objekt an einen solchen Parameter übergeben werden kann, aber nicht muss.	

Beispiele

- Beispiel 1 — PERFORM als interne Subroutine
- Beispiel 2 — PERFORM als externe Subroutine

Beispiel 1 — PERFORM als interne 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
```

Ausgabe des Programms 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
```

Beispiel 2 — PERFORM als externe Subroutine

Programm, das das PERFORM-Statement enthält:

```
** 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
```

Externe Subroutine PEREX3 mit vom Programm PEREX2 aufgerufenen Parametern:

```
** 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
```

Ausgabe des Programms 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
```