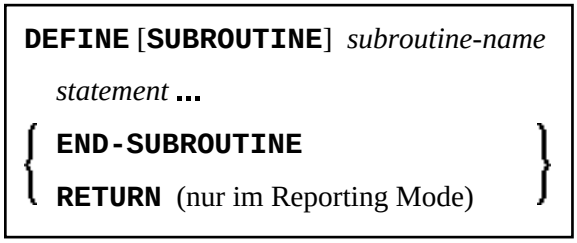
DEFINE SUBROUTINE



Dieses Kapitel behandelt folgende Themen:

- Funktion
- Einschränkungen
- Syntax-Beschreibung
- Welche Daten einer Subroutine zur Verfügung stehen
- Beispiele

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

Verwandte Statements: CALL | CALL FILE | CALL LOOP | CALLNAT | ESCAPE | FETCH | PERFORM

Gehört zur Funktionsgruppe: *Aufrufen von Programmen und Unterprogrammen*

Funktion

Das Statement DEFINE SUBROUTINE dient dazu, eine Natural-Subroutine zu definieren. Aufgerufen wird eine Subroutine mit einem PERFORM-Statement.

Interne und externe Subroutinen

Eine Subroutine kann entweder innerhalb des Natural-Objekts definiert werden, das das sie aufrufende PERFORM-Statement enthält (interne Subroutine); oder sie kann in einem anderen Natural-Objekt definiert werden als dem, welches das aufrufende PERFORM-Statement enthält (externe Subroutine). Eine interne Subroutine kann entweder vor oder nach dem ersten PERFORM-Statement, mit dem sie aufgerufen wird, definiert werden.

Anmerkung:
Die Verwendung externer Subroutinen empfiehlt sich zwar, um eine klar gegliederte Anwendungsstruktur zu erhalten; allerdings verursachen externe Subroutinen einen Verarbeitungsmehraufwand. Daher sollten nur größere funktionale Blöcke in externen Subroutinen untergebracht werden.

Einschränkungen

- Eine in einer Subroutine initiierte Verarbeitungsschleife muss vor dem `END-SUBROUTINE`-Statement wieder geschlossen werden.
- Eine interne Subroutine darf ihrerseits kein weiteres `DEFINE SUBROUTINE`-Statement enthalten (siehe *Beispiel 1* unten).
- Eine externe Subroutine (d.h. ein Objekt vom Typ Subroutine) darf nicht mehr als einen `DEFINE SUBROUTINE`-Statement-Block enthalten (siehe Beispiel 2 unten). Ein externer `DEFINE SUBROUTINE`-Block darf jedoch seinerseits interne Subroutinen enthalten (siehe *Beispiel 1* unten).

Beispiel 1

Die folgende Konstruktion ist in einem Objekt vom Typ Subroutine möglich, aber nicht in einem anderen Objekt (in dem `SUBR01` als interne Subroutine gälte):

```
...
DEFINE SUBROUTINE SUBR01
   ...
   PERFORM SUBR02
   PERFORM SUBR03
   ...
   DEFINE SUBROUTINE SUBR02
   /* inline subroutine...
   END-SUBROUTINE
 ...
   DEFINE SUBROUTINE SUBR03
   /* inline subroutine...
   END-SUBROUTINE
END-SUBROUTINE
END
```

Beispiel 2 (ungültig):

Die folgende Konstruktion ist in einem Objekt vom Typ Subroutine *nicht* erlaubt:

```
...
DEFINE SUBROUTINE SUBR01
...
END-SUBROUTINE
DEFINE SUBROUTINE SUBR02
...
END-SUBROUTINE
END
```

Syntax-Beschreibung

subroutine-name	Für den Namen einer Subroutine (maximal 32 Zeichen lang) gelten die gleichen Namenskonventionen wie für Benutzervariablen (siehe Abschnitt *Namen von Benutzervariablen* in der Dokumentation *Natural Studio benutzen*. Der Name einer Subroutine ist unabhängig vom Namen des Moduls, in dem sie definiert ist (beide Namen können, müssen aber nicht, gleich sein).
END-SUBROUTINE	Die Definition einer Subroutine wird mit `END-SUBROUTINE` beendet.
RETURN	Im Reporting Mode darf eine Subroutine auch mit `RETURN` beendet werden.

Welche Daten einer Subroutine zur Verfügung stehen

Dieses Abschnitt behandelt folgende Themen:

- Interne Subroutinen
- Externe Subroutinen

Interne Subroutinen

An eine interne Subroutine können mit dem `PERFORM`-Statement keine Parameter vom aufrufenden Programm übergeben werden.

Eine interne Subroutine kann auf die aktuelle Global Data Area sowie die vom aufrufenden Programm verwendete Local Data Area zugreifen.

Externe Subroutinen

Eine externe Subroutine kann auf die aktuelle Global Data Area zugreifen. Außerdem können Sie mit dem `PERFORM`-Statement Parameter direkt vom aufrufenden Objekt an die externe Subroutine übergeben; dadurch können Sie die Größe der Global Data Area klein halten.

Eine externe Subroutine kann nicht auf die im aufrufenden Programm definierte Local Data Area zugreifen; allerdings kann eine externe Subroutine eine eigene Local Data Area haben.

Beispiele

- Beispiel 1 — Subroutine definieren
- Beispiel 2 — Beispiel-Struktur für externe Subroutine mittels GDA-Feldern

Beispiel 1 — Subroutine definieren

```
** Example 'DSREX1S': DEFINE SUBROUTINE (structured mode)
************************************************************************
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
*
FORMAT PS=20
LIMIT 5
FIND EMPLOY-VIEW WITH NAME = 'SMITH'
  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 DSREX1S:

```
SMITH                    SMITH                    SMITH
ENGLANDSVEJ 222          3152 SHETLAND ROAD       14100 ESWORTHY RD.
                         MILWAUKEE                MONTERREY
554349                   877-4563                 994-2260

SMITH                    SMITH
5 HAWTHORN               13002 NEW ARDEN COUR
OAK BROOK                SILVER SPRING
150-9351                 639-8963
```

Äquivalentes Reporting-Mode-Beispiel: DSREX1R.

Beispiel 2 — Beispiel-Struktur für externe Subroutine mittels GDA-Feldern

```
** Example 'DSREX2': DEFINE SUBROUTINE (using GDA fields)
************************************************************************
DEFINE DATA
GLOBAL
  USING DSREX2G
END-DEFINE
*
INPUT 'Enter value in GDA field' GDA-FIELD1
*
*  Call external subroutine in DSREX2S
```

```
*
PERFORM DSREX2-SUB
*
END
```

Vom Programm DSREX2 benutzte Global Data Area DSREX2G:

```
1 GDA-FIELD1                        A    2
```

Vom Programm DSREX2 aufgerufene Subroutine DSREX2S:

```
** Example 'DSREX2S': SUBROUTINE (external subroutine using global data)
************************************************************************
DEFINE DATA
GLOBAL
  USING DSREX2G
END-DEFINE
*
DEFINE SUBROUTINE DSREX2-SUB
*
  WRITE 'IN SUBROUTINE' *PROGRAM '=' GDA-FIELD1
*
END-SUBROUTINE
*
END
```