DEFINE FUNCTION

```
DEFINE FUNCTION function-name
  [return-data-definition]
  [function-data-definition]
  statement...
END-FUNCTION
```

Dieses Kapitel behandelt folgende Themen:

- Funktion
- Syntax-Beschreibung
- Beispiel

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

Funktion

Mit dem `DEFINE FUNCTION`-Statement können Sie benutzerdefinierte Funktionen erstellen, die in den Natural-Statements anstelle von Operanden aufgerufen werden können. Diese Funktionen können nur innerhalb eines Natural-Objekts vom Typ Function definiert werden.

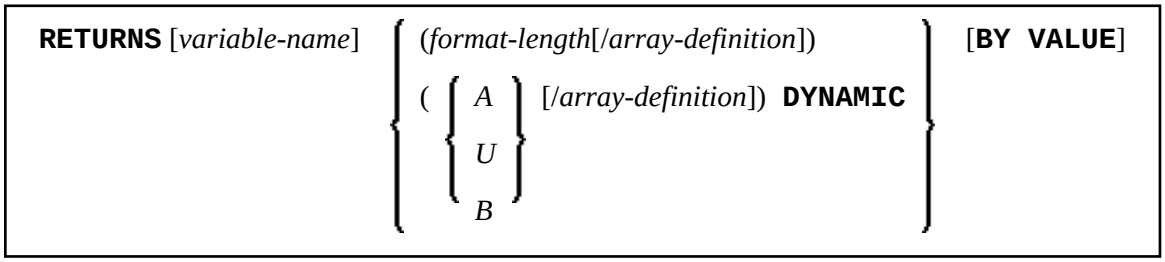
Weitere Informationen finden Sie in den folgenden Abschnitten im *Leitfaden zur Programmierung*:

- Natural-Objecttyp Function
- *Function Call*
- *Benutzerdefinierte Funktionen*

Syntax-Beschreibung

function-name	*function-name* ist der symbolische Name der zu definierenden Natural-Funktion. Es gelten die im Kapitel *Namenskonventionen für Benutzervariablen* in der Dokumentation *Natural Studio benutzen* aufgeführten Regeln. Das bedeutet, dass der Name maximal 32 Zeichen lang sein und mit einem Buchstaben oder einem Sonderzeichen, z.B. Rautensymbol (#), beginnen darf. Sie dürfen denselben Function-Namen nicht zweimal in einer Library benutzen (einschließlich der Libraries mit dem Steplib-Mechanismus). Funktionsüberladung ist nicht erlaubt. Dies bedeutet, dass alle Funktionsdefinitionen eindeutige Function-Namen haben müssen.
return-data-definition	Siehe *Rückgabedatendefinition* weiter unten.
function-data-definition	Siehe *Funktionsdatendefinition* weiter unten.
END-FUNCTION	Das für Natural reservierte Wort `END-FUNCTION` muss zum Beenden des `DEFINE FUNCTION`-Statements benutzt werden.

Rückgabedatendefinition

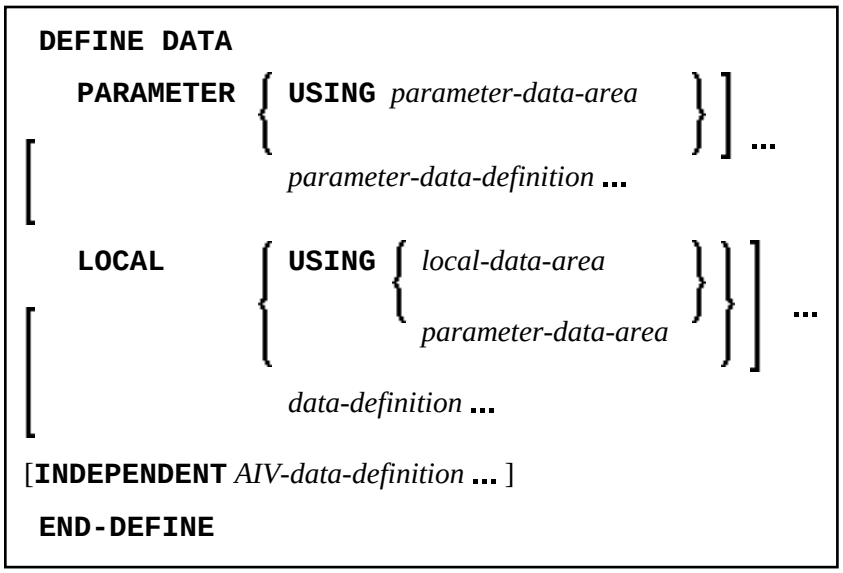


Syntax-Element-Beschreibung:

RETURNS	Jede Function darf nur eine Definition der Rückgabevariablen enthalten; d.h. es ist nur *eine* RETURNS-Klausel zulässig.
variable-name	Der Rückgabewert kann mit *variable-name* zugewiesen werden. Ist in der Definition kein expliziter Variablenname angegeben, wird der Name der Function als Rückgabevariable verwendet. Der Rückgabewert darf kein Array sein.
BY VALUE	Jeder Parameter kann direkt als Wert (By-Value) oder referenziert über seine Adresse (By-Reference) definiert werden, so dass es möglich ist, Werte über Parameter an den Caller zurückzugeben. Innerhalb einer Funktionsdefinition können rekursive Funktionsaufrufe verwendet werden. Wenn Sie das Schlüsselwort BY VALUE in der RETURNS-Klausel verwenden, wird der Rückgabewert der Function in das/die durch die RETURNS-Klausel festgelegte Format/Länge (*format-length*) umgesetzt.
format-length	Wenn Sie das Schlüsselwort BY VALUE weglassen, muss die Angabe für *format-length* bei der RETURNS-Klausel mit der von der zur Laufzeit ausgewerteten Funktion zurückgegebenen Format/Länge-Angabe übereinstimmen.
array-definition	Mit *array-definition* legen Sie die untere und obere Grenze einer Dimension bei einer Array-Definition fest. Weitere Informationen siehe DEFINE DATA-Statement, *Definition von Array-Dimensionen*.
DYNAMIC	Ein Parameter kann als DYNAMIC definiert werden. Informationen zur Verarbeitung von dynamischen Variablen siehe *Dynamische Variablen*.

Funktionsdatendefinition

Jedes Objekt des Typs Function darf nur eine Funktionsdatendefinition enthalten.



Wenn eine Function ein anderes Natural-Objekt aufruft, das eine Global Data (GDA) Area benutzt, dann erstellt sie ihre eigene GDA. Darum ist es nicht möglich, die aktuellen GDA-Daten des aufrufenden Objekts zu verändern. In der Function darf keine GDA angegeben werden.

Beispiel

Objekt vom Typ Function mit Funktionsdefinition:

```
DEFINE FUNCTION GET-FIRST-BYTE
  RETURNS (A1)
  DEFINE DATA PARAMETER
    1 #PARA (A10)
  END-DEFINE
  GET-FIRST-BYTE := #PARA /* return value is assigned
END-FUNCTION
END
```