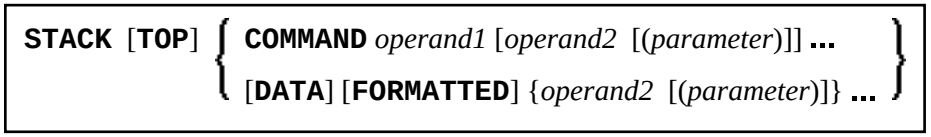
STACK



Dieses Kapitel behandelt folgende Themen:

- Funktion
- Syntax-Beschreibung
- Beispiel

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

Verwandte Statements: INPUT | RELEASE

Funktion

Das Statement `STACK` dient dazu, Daten im Natural-Stack abzulegen. Hierbei kann es sich um folgende Daten handeln:

- den Namen eines Natural-Programms oder -Systemkommandos, das ausgeführt werden soll;
- Daten, die bei der Ausführung eines `INPUT`-Statements als Eingabedaten verwendet werden sollen.

Weitere Informationen zum Stack finden Sie im Kapitel *Weitere Programmieraspekte*, *Stack* im *Leitfaden zur Programmierung*.

Syntax-Beschreibung

Operanden-Definitionstabelle:

Operand	Mögliche Struktur					Mögliche Formate													Referenzierung erlaubt	Dynam. Definition
operand1	C	S	A	G	N	A													ja	ja
operand2	C	S	A	G	N	A	U	N	P	I	F	B	D	T	L		G		ja	ja

Syntax-Element-Beschreibung:

TOP	Normalerweise werden die Daten unten im Stack abgelegt. Das Schlüsselwort `TOP` bewirkt, dass die Daten oben auf dem Natural Stack abgelegt werden. Beispiel: Mit diesem Statement wird der Inhalt der Variablen `#FIELDA` oben auf dem Stack abgelegt: `STACK TOP #FIELDA`

DATA	Mit `DATA` (Standardeinstellung) legen Sie Daten im Stack ab, die von einem `INPUT`-Statement als Eingabedaten verwendet werden.
	
	Begrenzungszeichen oder *Input Assign*-Zeichen innerhalb der übergebenen Datenwerte werden als Begrenzung interpretiert und entsprechend verarbeitet. Einzelheiten darüber, wie die Daten von einem `INPUT`-Statement verarbeitet werden, können Sie der Beschreibung des `INPUT`-Statements *Eingabedaten aus dem Natural-Stack* entnehmen.
	
	Beispiel: Mit den folgenden Statements wird der Inhalt der Variablen `#FIELD1` und `#FIELD2` im Stack abgelegt:
	
	`MOVE 'ABC' TO #FIELD1`
	`MOVE 'XYZ' TO #FIELD2`
	`STACK #FIELD1 #FIELD2`
	
	Diese Variablen werden als Eingabedaten an das nächste `INPUT`-Statement im Natural-Programm übergeben, und zwar im Begrenzungs-Modus:
	
	`INPUT #FIELD1  #FIELD2`
	
	Anmerkung:
	Wenn *operand2* eine Zeitvariable (Format T) ist, wird nur die Zeitkomponente des Variableninhalts im Stack abgelegt, aber nicht die Datumskomponente.

FORMATTED	Das Schlüsselwort FORMATTED bewirkt, dass alle Daten Feld für Feld an das nächste INPUT-Statement übergeben werden. Schlüsselzuordnungen oder Begrenzungszeichen werden nicht als solche interpretiert.
	
	Beispiele:
	
	Mit den folgenden Statements wird ABC,DEF in #FIELD1 übertragen und XYZ in #FIELD2:
	
	`MOVE 'ABC,DEF' TO #FIELD1`
	`MOVE 'XYZ'  TO #FIELD2`
	`STACK TOP DATA FORMATTED #FIELD1 #FIELD2`
	`...`
	`INPUT #FIELD1 #FIELD2`
	
	Angenommen, das Input-Begrenzungszeichen ist das Komma (Profil-/Session-Parameter ID=,), dann wird mit folgenden Statements — ohne das Schlüsselwort FORMATTED — ABC in #FIELD1 übertragen und DEF in #FIELD2:
	
	`MOVE 'ABC,DEF' TO #FIELD1`
	`STACK TOP DATA #FIELD1`
	`...`
	`INPUT #FIELD1 #FIELD2`
	
	Anmerkung:
	Die FORMATTED-Option sollte verwendet werden, wenn die zu übergebenden Daten Begrenzungs-, Steuer- oder DBCS-Zeichen enthalten, um eine unbeabsichtigte Interpretation dieser Zeichen zu vermeiden.

COMMAND *operand1*	Um ein Kommando (bzw. einen Programmnamen) im Stack abzulegen, geben Sie das Schlüsselwort COMMAND gefolgt von dem betreffenden Kommando (*operand1*) an. Würde ein Programm normalerweise den Benutzer mit einer Eingabeaufforderung in Form einer NEXT-Zeile konfrontieren, so unterdrückt nun Natural die Anzeige der NEXT-Zeile und führt stattdessen das im Stack abgelegte Kommando aus. Beispiel: Mit dem folgenden Statement wird das Kommando RUN oben auf dem Stack abgelegt. Natural führt dieses Kommando aus, wenn normalerweise das nächstemal die NEXT-Zeile ausgegeben würde: `STACK TOP COMMAND 'RUN'`

COMMAND *operand1 operand2 ...*	Zusammen mit einem Kommando (*operand1*) können Sie auch Daten (*operand2*) im Stack ablegen. Diese Daten werden dann vom nächsten INPUT-Statement nach der Ausführung des Kommandos als Eingabedaten verarbeitet. Zusammen mit einem Kommando abgelegte Daten werden immer unformatiert abgelegt. **Anmerkung:** Wenn in den abzulegenden Daten leere alphanumerische Felder (d.h. Leerzeichen) enthalten sind, werden diese Leerzeichen als Delimiter zwischen Werten interpretiert und folglich von dem betreffenden INPUT-Statement falsch verarbeitet. Wenn Sie daher leere alphanumerische Felder als Daten zusammen mit einem Kommando im Stack ablegen möchten, müssen Sie hierzu zwei STACK-Statements verwenden: Ein STACK DATA *operand2...*, um die Daten abzulegen, und ein STACK COMMAND *operand1*, um das Kommando abzulegen.
parameter	Wenn *operand2* eine Datumsvariable ist, können Sie den Session-Parameter DF (Datumsformat) als *parameter* für diese Variable angeben.

Beispiel

```
** Example 'STKEX1': STACK
************************************************************************
DEFINE DATA LOCAL
1 #CODE (A1)
END-DEFINE
*
INPUT //
  10X 'PLEASE SELECT COMMAND' //
  10X 'LIST VIEW      (V)' /
  10X 'LIST PROGRAM * (P)' /
  10X 'TECH INFO      (T)' /
  10X 'STOP           (.)' //
  20X 'CODE:' #CODE
*
*
DECIDE ON FIRST #CODE
  VALUE 'V'
    STACK TOP DATA    'VIEW'
    STACK TOP COMMAND 'LIST'
  VALUE 'P'
    STACK TOP COMMAND 'LIST PROGRAM *'
  VALUE 'T'
    STACK TOP COMMAND 'LAST *'
    STACK TOP COMMAND 'TECH'
    STACK TOP COMMAND 'SYSPROD'
  VALUE '.'
    STOP
  NONE
    REINPUT 'PLEASE ENTER VALID CODE'
```

```
END-DECIDE
*
*
END
```

Ausgabe des Programms STKEX1:

```
PLEASE SELECT COMMAND

LIST VIEW      (V)
LIST PROGRAM * (P)
TECH INFO      (T)
STOP           (.)

          CODE:P
```

Nach Eingabe und Bestätigung des Codes:

```
16:46:28               ***** NATURAL LIST COMMAND *****              2005-01-19
User HTR                 - LIST Objects in a Library -          Library SYSEXSYN

Cmd  Name       Type         S/C  SM Version  User ID    Date         Time
---  *_________ P__________  *__  *  *______  *________  *__________ *________
__   ACREX1     Program      S/C  S  4.1.03   RKE        2004-11-11  16:32:37
__   ACREX2     Program      S/C  S  4.1.03   RKE        2005-01-05  10:29:51
__   ADDEX1     Program      S/C  S  4.1.03   RKE        2004-11-11  16:36:49
__   AEDEX1R    Program      S/C  R  4.1.03   RKE        2004-11-11  16:40:34
__   AEDEX1S    Program      S/C  S  4.1.03   RKE        2004-11-11  16:39:57
__   AEPEX1R    Program      S/C  R  4.1.03   RKE        2004-11-11  16:41:57
__   AEPEX1S    Program      S/C  S  4.1.03   RKE        2004-11-11  16:42:31
__   AEPEX2     Program      S/C  S  4.1.03   RKE        2004-11-11  16:43:37
__   ASDEX1R    Program      S/C  R  4.1.03   RKE        2004-11-11  17:00:21
__   ASDEX1S    Program      S/C  S  4.1.03   RKE        2004-11-11  17:00:50
__   ASGEX1R    Program      S/C  R  4.1.03   RKE        2004-11-11  17:02:01
__   ASGEX1S    Program      S/C  S  4.1.03   RKE        2004-11-11  17:02:08
__   ATBEX1R    Program      S/C  R  4.1.03   RKE        2004-11-11  17:03:18
__   ATBEX1S    Program      S/C  S  4.1.03   RKE        2004-11-11  17:03:05
                                                                 14 Objects found
Top of List.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Print Exit  Sort        --    -     +     ++          >     Canc
```