User-Defined Constants

Constants can be used throughout Natural programs. This document discusses the types of constants that are supported and how they are used.

This chapter covers the following topics:

- Numeric Constants

- Alphanumeric Constants

- Unicode Constants

- Date and Time Constants

- Hexadecimal Constants

- Logical Constants

- Floating Point Constants

- Attribute Constants

- Handle Constants

- Defining Named Constants

Numeric Constants

The following topics are covered below:

- Numeric Constants

- Validation of Numeric Constants

Numeric Constants

A numeric constant may contain 1 to 29 numeric digits, a special character for decimal point notation (period or comma) and a sign.

Examples:

```
1234   +1234   -1234

12.34  +12.34  -12.34

MOVE 3 TO #XYZ
COMPUTE #PRICE = 23.34
COMPUTE #XYZ = -103
COMPUTE #A = #B * 6074
```

Numeric constants are represented internally in packed form (Format P); exception: if a numeric constant is used in an arithmetic operation in which the other operand is an integer variable (Format I), the numeric constant is represented in integer form (Format I).

Validation of Numeric Constants

When numeric constants are used within one of the statements COMPUTE, MOVE, or DEFINE DATA with INIT option, Natural checks at compilation time whether a constant value fits into the corresponding field. This avoids runtime errors in situations where such an error condition can already be detected during compilation.

Alphanumeric Constants

The following topics are covered below:

- Alphanumeric Constants

- Apostrophes Within Alphanumeric Constants

- Concatenation of Alphanumeric Constants

Alphanumeric Constants

An alphanumeric constant may contain 1 to 1 1073741824 bytes (1 GB) of alphanumeric characters.

An alphanumeric constant must be enclosed in either apostrophes (')

```
'text'
```

or quotation marks (")

```
"text"
```

Examples:

```
MOVE 'ABC' TO #FIELDX
MOVE '% INCREASE' TO #TITLE
DISPLAY "LAST-NAME" NAME
```

Note:
An alphanumeric constant that is used to assign a value to a user-defined variable must not be split between statement lines.

Apostrophes Within Alphanumeric Constants

If you want an apostrophe to be part of an alphanumeric constant that is enclosed in apostrophes, you must write this as two apostrophes or as a single quotation mark.

If you want an apostrophe to be part of an alphanumeric constant that is enclosed in quotation marks, you write this as a single apostrophe.

Example:

If you want the following to be output:

```
HE SAID, 'HELLO'
```

you can use any of the following notations:

```
WRITE 'HE SAID, ''HELLO'''
WRITE 'HE SAID, "HELLO"'
WRITE "HE SAID, ""HELLO"""
WRITE "HE SAID, 'HELLO'"
```

Note:
If quotation marks are not converted to apostrophes as shown above, this is due to the setting of the profile parameter TQ (Translate Quotation Marks); ask your Natural administrator for details.

Concatenation of Alphanumeric Constants

Alphanumeric constants may be concatenated to form a single value by use of a hyphen.

Examples:

```
MOVE 'XXXXXX' - 'YYYYYY' TO #FIELD
MOVE "ABC" - 'DEF' TO #FIELD
```

In this way, alphanumeric constants can also be concatenated with hexadecimal constants.

Unicode Constants

The following topics are covered below:

- Unicode Text Constants

- Apostrophes Within Unicode Text Constants

- Unicode Hexadecimal Constants

- Concatenation of Unicode Constants

Unicode Text Constants

A Unicode text constant must be preceded by the character U and enclosed in either apostrophes (')

```
U'text'
```

or quotation marks (")

```
U"text"
```

Example:

```
U'HELLO'
```

The compiler stores this text constant in the generated program in Unicode format (UTF-16).

Apostrophes Within Unicode Text Constants

If you want an apostrophe to be part of a Unicode text constant that is enclosed in apostrophes, you must write this as two apostrophes or as a single quotation mark.

If you want an apostrophe to be part of a Unicode text constant that is enclosed in quotation marks, you write this as a single apostrophe.

Example:

If you want the following to be output:

```
HE SAID, 'HELLO'
```

you can use any of the following notations:

```
WRITE U'HE SAID, ''HELLO'''
WRITE U'HE SAID, "HELLO"'
WRITE U"HE SAID, ""HELLO"""
WRITE U"HE SAID, 'HELLO'"
```

Note:
If quotation marks are not converted to apostrophes as shown above, this is due to the setting of the profile parameter TQ (Translate Quotation Marks); ask your Natural administrator for details.

Unicode Hexadecimal Constants

The following syntax is used to supply a Unicode character or a Unicode string by its hexadecimal notation:

```
UH'hhhh...'
```

where *h* represents a hexadecimal digit (0-9, A-F). Since a UTF-16 Unicode character consists of a double-byte, the number of hexadecimal characters supplied has to be a multiple of four.

Example:

This example defines the string 45.

```
UH'00340035'
```

Concatenation of Unicode Constants

Concatenation of Unicode text constants (U) and Unicode hexadecimal constants (UH) is allowed.

Valid Example:

```
MOVE U'XXXXXX' - UH'00340035' TO #FIELD
```

Unicode text constants or Unicode hexadecimal constants cannot be concatenated with code page alphanumeric constants or H constants.

Invalid Example:

```
MOVE U'ABC' - 'DEF' TO #FIELD
MOVE UH'00340035' - H'414243' TO #FIELD
```

Further Valid Example:

```
DEFINE DATA LOCAL
1 #U10 (U10)            /* Unicode variable with 10 (UTF-16) characters, total byte length = 20
1 #UD  (U)  DYNAMIC     /* Unicode variable with dynamic length
END-DEFINE
*
#U10 := U'ABC'          /* Constant is created as X'004100420043' in the object, the UTF-16 representation for string 'ABC'.

#U10 := UH'004100420043'   /* Constant supplied in hexadecimal format only, corresponds to U'ABC'

#U10 := U'A'-UH'0042'-U'C' /* Constant supplied in mixed formats, corresponds to U'ABC'.
END
```

Date and Time Constants

The following topics are covered below:

- Date Constant

- Time Constant

- Extended Time Constant

Date Constant

A date constant may be used in conjunction with a Format D variable.

Date constants may have the following formats:

D'yyyy-mm-dd'	International date format
D'dd.mm.yyyy'	German date format
D'dd/mm/yyyy'	European date format
D'mm/dd/yyyy'	US date format

where dd represents the number of the day, mm the number of the month and $yyyy$ the year.

Example:

```
DEFINE DATA LOCAL
1 #DATE (D)
END-DEFINE
...
MOVE D'2004-03-08' TO #DATE
...
```

The default date format is controlled by the profile parameter DTFORM (Date Format) as set by the Natural administrator.

Time Constant

A time constant may be used in conjunction with a Format T variable.

A time constant has the following format:

T'*hh:ii:ss*'

where *hh* represents hours, *ii* minutes and *ss* seconds.

Example:

```
DEFINE DATA LOCAL
1 #TIME (T)
END-DEFINE
...
MOVE T'11:33:00' TO #TIME
...
```

Extended Time Constant

A time variable (Format T) can contain date and time information, date information being a subset of time information; however, with a "normal" time constant (prefix T) only the time information of a time variable can be handled:

T'*hh:ii:ss*'

With an extended time constant (prefix E), it is possible to handle the full content of a time variable, including the date information:

E'*yyyy-mm-dd hh:ii:ss*'

Apart from that, the use of an extended time constant in conjunction with a time variable is the same as for a normal time constant.

Note:
The format in which the date information has to be specified in an extended time constant depends on the setting of the profile parameter DTFORM. The extended time constant shown above assumes DTFORM=I (international date format).

Hexadecimal Constants

The following topics are covered below:

- Hexadecimal Constants

- Concatenation of Hexadecimal Constants

Hexadecimal Constants

A hexadecimal constant may be used to enter a value which cannot be entered as a standard keyboard character.

A hexadecimal constant may contain 1 to 1073741824 bytes (1 GB) of alphanumeric characters.

A hexadecimal constant is prefixed with an H. The constant itself must be enclosed in apostrophes and may consist of the hexadecimal characters 0 - 9, A - F. Two hexadecimal characters are required to represent one byte of data.

The hexadecimal representation of a character varies, depending on whether your computer uses an ASCII or EBCDIC character set. When you transfer hexadecimal constants to another computer, you may therefore have to convert the characters.

ASCII examples:

```
H'313233'    (equivalent to the alphanumeric constant '123')
H'414243'    (equivalent to the alphanumeric constant 'ABC')
```

EBCDIC examples:

```
H'F1F2F3'    (equivalent to the alphanumeric constant '123')
H'C1C2C3'    (equivalent to the alphanumeric constant 'ABC')
```

When a hexadecimal constant is transferred to another field, it will be treated as an alphanumeric value (format A).

The data transfer of an alphanumeric value (format A) to a field which is defined with a format other than A,U or B is not allowed. Therefore, a hexadecimal constant used as initial value in a DEFINE DATA statement is rejected with the syntax error NAT0094 if the corresponding variable is not of format A, U or B.

Example:

```
DEFINE DATA LOCAL
1 #I(I2) INIT <H'000F'>       /* causes a NAT0094 syntax error
END-DEFINE
```

Concatenation of Hexadecimal Constants

Hexadecimal constants may be concatenated by using a hyphen between the constants.

ASCII example:

```
H'414243' - H'444546' (equivalent to 'ABCDEF')
```

EBCDIC example:

```
H'C1C2C3' - H'C4C5C6' (equivalent to 'ABCDEF')
```

In this way, hexadecimal constants can also be concatenated with alphanumeric constants.

Logical Constants

The logical constants TRUE and FALSE may be used to assign a logical value to a field defined with Format L.

Example:

```
DEFINE DATA LOCAL
1 #FLAG (L)
END-DEFINE
...
MOVE TRUE TO #FLAG
...
IF #FLAG ...
  statement ...
  MOVE FALSE TO #FLAG
END-IF
 ...
```

Floating Point Constants

Floating point constants can be used with variables defined with format F.

Example:

```
DEFINE DATA LOCAL
1 #FLT1 (F4)
END-DEFINE
...
COMPUTE #FLT1 = -5.34E+2
...
```

Attribute Constants

Attribute constants can be used with variables defined with format C (control variables). This type of constant must be enclosed within parentheses.

The following attributes may be used:

Attribute	Description
`AD=D`	default
`AD=B`	blinking
`AD=I`	intensified
`AD=N`	non-display
`AD=V`	reverse video
`AD=U`	underlined
`AD=C`	cursive/italic
`AD=Y`	dynamic attribute
`AD=P`	protected
`CD=BL`	blue
`CD=GR`	green
`CD=NE`	neutral
`CD=PI`	pink
`CD=RE`	red
`CD=TU`	turquoise
`CD=YE`	yellow

See also session parameters `AD` and `CD`.

Example:

```
DEFINE DATA LOCAL
1 #ATTR (C)
1 #FIELD (A10)
END-DEFINE
...
MOVE (AD=I CD=BL) TO #ATTR
...
INPUT #FIELD (CV=#ATTR)
...
```

Handle Constants

The handle constant `NULL-HANDLE` can be used with object handles.

For further information on object handles, see the section *NaturalX*.

Defining Named Constants

If you need to use the same constant value several times in a program, you can reduce the maintenance effort by defining a named constant:

- Define a field in the DEFINE DATA statement,

- assign a constant value to it, and

- use the field name in the program instead of the constant value.

Thus, when the value has to be changed, you only have to change it once in the DEFINE DATA statement and not everywhere in the program where it occurs.

You specify the constant value in angle brackets with the keyword CONSTANT after the field definition in the DEFINE DATA statement.

- If the value is alphanumeric, it must be enclosed in apostrophes.

- If the value is text in Unicode format, it must be preceded by the character U and must be enclosed in apostrophes.

- If the value is in hexadecimal Unicode format, it must be preceded by the characters UH and must be enclosed in apostrophes.

Example:

```
DEFINE DATA LOCAL
1 #FIELDA (N3) CONSTANT <100>
1 #FIELDB (A5) CONSTANT <'ABCDE'>
1 #FIELDC (U5) CONSTANT <U'ABCDE'>
1 #FIELDD (U5) CONSTANT <UH'00410042004300440045'>
END-DEFINE
...
```

During the execution of the program, the value of such a named constant cannot be modified.