関数

関数について

関数は、データ型を変換したり、文字列や日時値、時間値などを使用した計算を行なうときに使用します。
関数を使用することで、表やレコードの情報、桐の動作環境などを調べることもできます。

分類

関数はつぎの分類に分けられます。

- データ型変換
- 算術
- 文字列操作
- 文字種
- 日時
- 集計
- 比較・判定
- 置換
- ファイル
- レコードの属性
- 表の属性
- 表の状態
- フォーム・レポート
- システム
- ネットワーク
- 変数操作
- ODBC（外部 DB）
- その他

戻り値

関数の戻り値には、つぎのものがあります。

- 文字列型
- 整数型
- 長整数型
- 数値型
- 実数型
- 日時型
- 時間型

各関数の戻り値の型は、本マニュアルではそれぞれの関数の標題の右側に記載します。

 # グループ選択状態 数値型
↑
戻り値の型

ただし、関数によって、引き数の型や内容で戻り値の型が変化するものがあります。そのような場合は「?」と表記し、詳細は各関数の説明に記載します。

引数

関数には、計算元の値や計算方法、表示書式などの引数を必要とするものがあります。本マニュアルでは、これらの引数の種類を、つぎの形式で表記しています。

表記	説明
str	計算結果が文字列型になる計算式を指定します。 文字列型の項目と変数も指定できます。
num	計算結果が数値になる計算式を指定します。 数値型、通貨型、整数型、長整数型、実数型の項目と変数も指定できます。
n	計算結果が数値になる計算式を指定します。 数値型、通貨型、整数型、長整数型、実数型の項目と変数も指定できます。 小数点以下の値は切り捨てられます。
tmst	計算結果が日時型になる計算式を指定します。 日時型の項目と変数も指定できます。
itvl	計算結果が時間型になる計算式を指定します。 時間型の項目と変数も指定できます。
val	代入先のデータ型と同じになる計算式を指定します。 たとえば計算結果を項目値に代入する場合は、その項目のデータ型と同じになる計算式を指定します。
sw	計算結果が 0 か 1 になる計算式を指定します。
f	計算結果が決められた番号になる数値型の計算式を指定します。 決められた番号以外の値を指定するとエラーになります。
item	項目の名前を指定します。

項目の名前の前後は[ ]でくくります。
変数の名前に、変数または計算式を指定することはできません。

var　変数名を指定します。
変数名の前には、& をつけます。
変数名を、別の変数または計算式で指定することはできません。

index　配列変数の要素番号を指定します。
要素番号を、変数または計算式で指定することはできません。

file　ファイル名を表わす文字列型の計算式を指定します。
ファイルの保存場所（パス）をつけないときは、現在のデータ パスが自動的に付加されます。
文字列型の項目と変数も指定できます。

op　その関数で決められたオプションを指定します。
変数名と計算式は、指定できません。

cond　条件式を指定します。
式の計算結果が 0 または未定義値なら偽、それ以外なら真と判断されます。
論理演算子（.and、.or、.not）が使用できます。

省略できる引数は、つぎの形式で表記しています。

（例）＃部分文字列(str , n1 【 | , n2 】)
→＃部分文字列(str , n1)
または
→＃部分文字列(str , n1 , n2)

いずれかひとつの内容を引数に指定する場合は、つぎの形式で表記しています。

（例）＃日付(【 str | tmst 】 , f)
→＃日付(str , f)
または
→＃日付(tmst , f)

データ型などに応じて、指定する引数の数が異なる場合は、つぎの形式で表記しています。

（例）＃時間値(【 str | num , n 】)
→＃時間値(str)
または
→＃時間値(num , n)

算術計算の精度

三角関数の計算精度は、使用しているコプロセッサの仕様に依存します。コプロセッサを使用しないときの計算精度は、約 6 桁です。

関数一覧

#ACOS (num) 数値型

数値 num の逆余弦を求めます。求めた値の単位はラジアンです。

■ノート

- 計算精度は、使用しているコプロセッサの仕様に依存します。コプロセッサを使用しないときの計算精度は6桁です。
- 外部データベースの表からデータを取り出すときは、ODBC 関数の ACOS(num) と同じ計算結果になります。

#ASIN (num) 数値型

数値 num の逆正弦を求めます。求めた値の単位はラジアンです。

■ノート

- 計算精度は、使用しているコプロセッサの仕様に依存します。コプロセッサを使用しないときの計算精度は6桁です。
- 外部データベースの表からデータを取り出すときは、ODBC 関数の ASIN(num) と同じ計算結果になります。

#ATAN (num) 数値型

数値 num の逆正接を求めます。求めた値の単位はラジアンです。

■ノート

- 計算精度は、使用しているコプロセッサの仕様に依存します。コプロセッサを使用しないときの計算精度は6桁です。
- 外部データベースの表からデータを取り出すときは、ODBC 関数の ATAN(num) と同じ計算結果になります。

#CASE (n , val1 , … , valn) ?

n 番目の式 val を実行し、計算結果を返します。

引数	説明
n	取り出す val の番号を指定します。 通常、ここには変数名や計算式を指定します。 この値が 1 未満か n より大きいときは未定義値になります。
vali	計算式を指定します。

■サンプル

- 現在の日時値に応じて、月の英語文字列を求めます。

```
#CASE(# 月(# 日時値),"January", "February","March","April","May",
"June","July","August","September","October","November","December")
```

#CEIL (num) 数値型

num 以上の最も小さい整数を求めます。

■サンプル

- 指定する数値以上の最も小さい整数を求めます。

```
#CEIL( −3.4 )    →  −3
#CEIL( 3.4 )     →  4
```

- [単価] を 500 円単位に丸めます。1 ～ 500 を 500 円、501 ～ 1,000 を 1,000 円に丸めます（参照： #FLOOR)。

```
#CEIL( [単価] / 500 ) * 500
```

■ノート

- 外部データベースの表からデータを取り出すときは、ODBC 関数の CEILING(num) と同じ計算結果になります。

#COS (num) 数値型

ラジアンを単位とする数値 num の余弦を求めます。

■ノート

- 計算精度は、使用しているコプロセッサの仕様に依存します。コプロセッサを使用しないときの計算精度は 6 桁です。
- 外部データベースの表からデータを取り出すときは、ODBC 関数の COS(num) と同じ計算結果になります。

#DISKFREE (str) 数値型

ドライブ str のディスクの空き容量をバイト単位の数値で返します。
str の先頭 1 文字をドライブ名として使用します。
ディスクの準備ができていないか、指定するドライブが存在しないときは −1 になります。

■サンプル

- A ドライブの空き容量を調べます。

```
#DISKFREE( "A" )  →  49152（byte）
```

- 編集対象表が保存されているドライブの空き容量を調べます。

```
#DISKFREE( # 表ファイル名( #IS 表 ) )
```

#DISKSIZE (str) 数値型

ドライブ str の全ディスク容量をバイト単位の数値で返します。
str の先頭 1 文字をドライブ名として使用します。
ディスクの準備ができていないか、指定するドライブが存在しないときは −1 になります。

#DotToInch (num) 数値型

dot 単位の num を、画面解像度に応じた inch 単位の数値に変換します。
求めた値は、Windows の [画面] プロパティで設定した画面解像度に応じて異なります（デスクトップの領域サイズに応じて変わるわけではありません）。
画面解像度が 96 dpi であれば、96 dot = 1 inch になります。

#DotToMili (num) 数値型

dot 単位の num を、画面解像度に応じた mm 単位の数値に変換します。
1mm は、1/25.4 inch として計算されます。
求めた値は、Windows の[画面]プロパティで設定した画面解像度に応じて異なります（デスクトップの領域サイズに応じて変わるわけではありません）。

#DotToPoint (num) 数値型

dot 単位の num を、画面解像度に応じた points 単位の数値に変換します。
1 points は、1/72 inch として計算されます。
求めた値は、Windows の[画面]プロパティで設定した画面解像度に応じて異なります（デスクトップの領域サイズに応じて変わるわけではありません）。

■サンプル

- 14 dot に、何 points のフォントサイズの文字が入るか調べます。一般的には、この計算で求めた値よりも −1 points 程度のフォントしか入りません。フォームのテキストオブジェクトなどでは、上下左右に 2 dot 以上（計 4 dot 以上）の余白を考慮する必要があります。

  #DotToPoint(14) → 10.5（points）

#EXP (num) 数値型

e の num 乗を求めます。

■ノート

- 計算精度は、使用しているコプロセッサの仕様に依存します。コプロセッサを使用しないときの計算精度は 6 桁です。
- 外部データベースの表からデータを取り出すときは、ODBC 関数の EXP(num) と同じ計算結果になります。

#FLOOR (num) 数値型

num 以下の最も大きい整数を求めます。

■サンプル

- 指定する数値以下の最も大きい整数を求めます。

  #FLOOR(−3.4) → −4
  #FLOOR(3.4) → 3

- [単価] を 500 円単位に丸めます。0 ～ 499 を 0 円、500 ～ 999 を 500 円に丸めます（参照：#CEIL）。

  #FLOOR([単価] / 500) * 500

■ノート

- 外部データベースの表からデータを取り出すときは、ODBC 関数の FLOOR(num) と同じ計算結果になります。

#GETENV (str) 文字列

str で指定した環境変数の値を取り出します。
指定した環境変数がなければ、未定義値を返します。

■サンプル

- Windows の起動場所を調べ、そのフォルダ名と「¥System¥Shell32.dll」を、ひとつの文字列につなげます。

```
#GETENV("winbootdir")+"¥System¥Shell32.dll"
→ C:¥WINDOWS¥System¥Shell32.dll
```

#HEX (【 str | num 】) ?

16 進数表記の文字列を数値に、数値を 16 進数表記の文字列に変換します。

引数	説明
str	数値に変換する 16 進数表記の文字列を指定します。 8 桁より長い 16 進数表記の文字列を指定すると、 8 桁を超える部分は、無視して変換します。
num	16 進数表記の文字列に変換する数値を指定します。

■サンプル

- 16 進数 4E5A を 数値に変換します。

```
#HEX( "4E5A" ) → 20058
```

- 13901 を 16 進数表記の文字列に変換します。

```
#HEX( 13901 ) → 364D
```

- 「桐」という文字を16 進数表記の文字列に変換します。

```
#HEX( #JIS( "桐" ) ) → 364D
```

- 16 進数 364D を 文字に変換します。

```
#JIS( #HEX( "364D " ) ) → 桐
```

■ノート

- num に指定できる数値は、−2147483647 ～ 2147483647 の範囲の整数です。
- str に指定する 16 進数は、最上位ビットを符号とする 32 ビット範囲内の文字列とします。
- #HEX 使用したときの負の数は、符号付きの 16 進数文字に変換されます。
- #HEX32 を使用したときの負の数は、2 の補数表現のビットとなる 16 進数文字に変換されます。
- #HEX は #H と書いてもかまいません。

#HEX32（【 str | num 】） ?

16 進数表記の文字列を数値に、数値を 16 進数表記の文字列に変換します。

引数	説明
str	数値に変換する 16 進数表記の文字列を指定します。 8 桁より長い 16 進数表記の文字列を指定すると、エラーになります。
num	16 進数表記の文字列に変換する数値を指定します。

■ノート

- num に指定できる数値は符号付きの 32 ビットで扱える範囲内の整数（−2147483648 ～ 2147483647）です。
- str に指定する 16 進数は、最上位ビットを符号とする 32 ビット範囲内の文字列とします。
- #HEX 使用したときの負の数は、符号付きの 16 進数文字に変換されます。
- #HEX32 を使用したときの負の数は、2 の補数表現のビットとなる 16 進数文字に変換されます。

#InchToDot（ num ） 数値型

inch 単位の num を、画面解像度に応じた dot 数に変換します。
求めた値は、Windows の［画面］プロパティで設定した画面解像度に応じて異なります（デスクトップの領域サイズに応じて変わるわけではありません）。

■サンプル

- 現在の画面解像度を求めます。

```
#InchToDot( 1 )　→　96（dpi = 96 dot / 1 inch）
```

#INITCAP（ str ） 文字列型

文字列 str 内の英単語の先頭文字を大文字に変換します。
空白文字を単語の区切りとし、大文字はそのまま返します。

■サンプル

- 英単語の先頭文字を大文字に変換します。英字の大文字は、そのままになります。

```
#INITCAP("SQL(=structured query language )") →　SQL(=Structured Query Language )
```

- すべての英字が大文字のときは、この関数を使っても効果がありません。英単語の先頭文字だけ大文字に変換するには、小文字に変換した文字列を、この関数の引数にします。

```
#INITCAP(#LC("STRUCTURED QUERY LANGUAGE"))　→　Structured Query Language
```

#INT（ num ） 数値型

num の整数部を取り出します。

■サンプル

- 34.567 の整数部を求めます。

```
#INT( 34.567 )　→　34
```

#ISLC (str , sw) 数値型

文字列の中に、小文字が含まれているかどうかを調べます。この関数で調べる小文字は、全角と半角の英字、ギリシア文字とロシア文字です。

引数	説明
str	調べる文字列を指定します。
sw	調べる方法を 1 か 0 で指定します。

値	戻り値
1	str がすべて小文字であれば真（1）、そうでなければ偽（0）を返します。
0	str に小文字が含まれていればその文字位置、含まれていなければ偽（0）を返します。

#ISREMOTE (str) 数値型

ドライブ str がリモートドライブかどうかを調べます。
リモートドライブのときは真（1）、ローカルドライブのときは偽（0）を返します。
str の先頭 1 文字をドライブ名として使用します。

#ISUC (str , sw) 数値型

文字列の中に、大文字が含まれているかどうかを調べます。この関数で調べる大文字は、全角と半角の英字、ギリシア文字とロシア文字です。

引数	説明
str	調べる文字列を指定します。
sw	調べる方法を 1 か 0 で指定します。

値	戻り値
1	str がすべて大文字であれば真（1）、そうでなければ偽（0）を返します。
0	str に大文字が含まれていればその文字位置、含まれていなければ偽（0）を返します。

#IS 英字 (str , sw) 数値型

文字列の中に、全角または半角の英字が含まれているかどうかを調べます。

引数	説明
str	調べる文字列を指定します。
sw	調べる方法を 1 か 0 で指定します。

値	戻り値
1	str がすべて英字であれば真（1）、そうでなければ偽（0）を返します。
0	str に英字が含まれていればその文字位置、含まれていなければ偽（0）を返します。