

?

word

Оператор вызова справки по языку.

'word

(accepts: any-type)

Синоним для справки.

? if

If condition is TRUE, evaluates the block.

Arguments:

condition --
block -- (block)

*

value1 value2

Оператор умножения.

value1

(accepts: char number money time tuple)

value2

(accepts: char number money time tuple)

print 123 * 10

1230

print 12.3 * 10

123

print 3:20 * 3

10:00

**

number number

Возвращает первое число, возведенное в степень вторым.

number

(accepts: number)

number

(accepts: number)

print 10 ** 2

100

print 12.345 ** 5

286718.338524636

+

value1 value2

Оператор сложения.

value1

(accepts: char number money date time tuple)

value2

(accepts: char number money date time tuple)

print 123 + 1

124

print 1.2.3.4 + 4.3.2.1

5.5.5.5

print 100.200.255.255 + 1.100.0.255

101.255.255.255

print 12:00 + 11:00

23:00

-

value1 value2

Оператор вычитания.

value1

(accepts: char number money date time tuple)

value2

(accepts: char number money date time tuple)

print 123 - 1

122

print -123 ; a negative number

-123

print - 123 ; negating a positive number (unary)

-123

print 1.2.3.4 - 1.0.3.0

0.2.0.4

```
print 12:00 - 11:00
1:00
```

/

value1 value2

Возвращает первое значение, разделенное на второе.

value1 (*accepts: char number money time tuple*)

value2 (*accepts: char number money time tuple*)

```
print 123 / 10
12.3
print 12.3 / 10
1.23
print 1:00 / 60
0:01
```

//

value1 value2

Возвращает остаток от первого значения, разделенного на секунду.

value1 (*accepts: char number money time tuple*)

value2 (*accepts: char number money time tuple*)

```
print 123 // 10
3
print 25:32 // 24:00
1:32
```

<

value value

Возвращает TRUE если первое значение меньше чем второе.

value (*accepts: any value*)

value (*accepts: any value*)

```
print "abc" < "abcd"
true
print 15-June-1999 < 14-June-1999
false
print 1.2.3.4 < 4.3.2.1
true
print 1:30 < 2:00
true
```

<=

value value

Возвращает TRUE, если первое значение меньше чем или равно второму значению.

value (*accepts: any value*)

value (*accepts: any value*)

```
print "abc" <= "abd"
true
print 14-June-1999 <= 15-June-1999
true
print 4.3.2.1 <= 1.2.3.4
false
print 1:23 <= 10:00
true
```

пробел

value value

Возвращает TRUE, если значения не равны.

value (*accepts: any value*)

value (*accepts: any value*)

```
print "abc" "abcd"
true
print [1 2 4] [1 2 3]
true
print 12-sep-98 10:30
true
```

=

value value

Возвращает TRUE, если значения равны.

value (accepts: any value)

value (accepts: any value)

```
print 123 = 123
true
print "abc" = "abc"
true
print [1 2 3] = [1 2 4]
false
print 15-June-1998 = 15-June-1999
false
print 1.2.3.4 = 1.2.3.0
false
print 1:23 = 1:23
true
```

=

value value

Возвращает TRUE, если значения равны и того же самого datatype.

value (accepts: any value)

value (accepts: any value)

```
print 123 == 123
true
print "abc" == "ABC"
false
```

=?

value value

Возвращает TRUE, если значения идентичны.

value (accepts: any value)

value (accepts: any value)

```
a: "apple"
b: a
print "Apple" =? "APPLE"
true
print a =? b
true
```

>

value value

Возвращает TRUE, если первое значение больше чем второе.

value (accepts: any value)

value (accepts: any value)

```
print "abc" > "abb"
true
print 16-June-1999 > 15-June-1999
true
print 4.3.2.1 > 1.2.3.4
true
print 11:00 > 12:00
false
```

>=

value value

Возвращает TRUE если первое значение больше чем или равно второму значению.

value (accepts: any value)

value (accepts: any value)

```
print "abc" >= "abb"
true
print 16-June-1999 >= 15-June-1999
true
print 1.2.3.4 >= 4.3.2.1
false
print 11:00 >= 11:00
true
```

about**Информация о REBOL**`now``about`

More information about the REBOL language can be obtained from <http://WWW.REBOL.COM>.

absolute*value***Возвращает абсолютное значение.****value** (*accepts: number money time*)`print absolute -123``123``print absolute -1:23``1:23`**add***value1 value2***Возвращает результат из добавления двух значений.****value1** (*accepts: char number money date time tuple*)**value2** (*accepts: char number money date time tuple*)`print add 123 1``124``print add 1.2.3.4 4.3.2.1``5.5.5.5``print add 3:00 -4:00``-1:00`**alias***word name***Создает запись по буквам замены для слова.****word** Word to alias (*accepts: word*)**name** Name of alias (*accepts: string*)

Alias создает слово синонима, которое является записью по буквам альтернативы для слова и идентично во всех отношениях (сравнение символа и ссылки в переменной). Alias должен быть сделан до загрузки файлов, содержащих новый символ. Не пугайте Alias с установкой другого слова к тому же самому значению. То есть P::Print - не тот же самый, поскольку псевдоним 'Печатает "P"'. Последний делает P и печать тем же самым словом во всех отношениях.

`alias 'print "stampa"``do {stampa "Mondo REBOL"}``Mondo REBOL`**all***block***Вычисляет и возвращает при первом FALSE или NONE.****block** Block of expressions (*accepts: block*)

Оценивает каждое выражение блока, на наличие другого значения кроме FALSE или NONE, тогда возвращает TRUE.

`a: 100``b: -100``if all [(a > b) (a > 10) (b < -3)] [print "all were true"]``all were true`**and***value1 value2***Возвращает первое значение ANDed со вторым.****value1** (*accepts: logic char number tuple*)**value2** (*accepts: logic char number tuple*)

Инфиксный оператор. Для ЛОГИЧЕСКИХ значений, оба должны быть TRUE, чтобы вернуть TRUE, иначе возвращает FALSE. Для целых чисел, рассматривает каждый бит

```

отдельно.
print 123 and 1
1
print true and true
true
print true and false
false
print (10 < 20) and (20 > 15)
true
print 1.2.3.4 and 255.0.255.0
1.0.3.0

```

any

block

Вычисляет и возвращает первое значение, которое не FALSE или NONE.

block

Block of expressions (*accepts: block*)

Оценивает каждое выражение блока и возвращает значение первого, который не является NONE или FALSE, иначе возвращает TRUE.

```

a: 100
b: -100
if any [(a > b) (a > 10) (b < -3)] [print "one was true"]
one was true

```

any-block!

Представляет ANY-BLOCK datatype, внешне и внутренне в пределах системы. Когда снабжено как параметр к функции MAKE, это определяет тип значения, которое будет создано. Когда обеспечено в пределах спецификации параметра функции, просит интерпретатор проверить что, значение параметра имеет указанный тип, когда функция вычислена.

any-block?

value

Возвращает TRUE для значения any-block.

value

(*accepts: any-type*)

Возвращает FALSE для всех других значений.

```

print any-block? [1 2]
true
print any-block? 12
false

```

any-function!

Возвращает значение, которое имеет тот же самый datatype, так что это может использоваться, чтобы MAKE новые значения того типа.

any-function?

value

Возвращает TRUE для значения any-function.

value

(*accepts: any-type*)

Возвращает FALSE для всех других значений.

```

print any-function? :find
true
print any-function? 123
false

```

any-string!

Когда снабжено как параметр к функции MAKE, это определяет тип значения, которое будет создано. Когда обеспечено в пределах спецификации параметра функции, просит интерпретатор проверить, что значение параметра имеет указанный тип, когда функция вычислена.

any-string?

value

Возвращает TRUE для значения any-string.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
if any-string? "Hello" [print "a string"]
a string
```

any-type!

Когда снабжено как параметр к функции MAKE, это определяет тип значения, которое будет создано. Когда обеспечено в пределах спецификации параметра функции, просит интерпретатор проверить, что значение параметра имеет указанный тип, когда функция вычислена.

any-type?

value

Возвращает TRUE для значения any-type.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print any-type? http://www.REBOL.com
true
```

any-word!

Когда снабжено как параметр к функции MAKE, это определяет тип значения, которое будет создано. Когда обеспечено в пределах спецификации параметра функции, просит интерпретатор проверить, что значение параметра имеет указанный тип, когда функция вычислена.

any-word?

value

Возвращает TRUE для значения any-word.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print any-word? 'word
true
```

append

series value

Добавляет к значению другое значение в конец строки.

series

(accepts: series port)

value

(accepts: any value)

```
test: [1 2 3 4]
append test [5 6]
print test
1 2 3 4 5 6
```

arccosine

value

Возвращает тригонометрический арккосинус в градусах.

value

(accepts: number)

/radians

Returns result in radians.

Обратная функция к косинусу.

```
print arccosine .5
60
```

arcsine

value

Возвращает тригонометрический арксинус в градусах.

value

(accepts: number)

/radians

Returns result in radians.

Обратная функция к синусу.

```
print arcsine .5
```

```
30
```

arctangent

value

Возвращает тригонометрический арктангенс в градусах.

value

(accepts: number)

/radians

Returns result in radians.

Обратная функция к тангенсу.

```
print arctangent .22
```

```
12.4074185274007
```

array

size

Создает и инициализирует ряд данного размера. Массив.

size

Size or block of sizes for each dimension *(accepts: integer block)*

/initial

Specify an initial value for all elements

value

Initial value *(accepts: any value)*

```
numbers: array 5 ; makes an array of 5 elements
```

```
image: array [10 25] ; makes an array of 10 rows of 25 values
```

```
xyz: array [10 25 5] ; makes a three-dimensional array
```

```
repeat index 5 [poke numbers index index]
```

```
print numbers
```

```
1 2 3 4 5
```

ask

question

Запрашивается для пользователя ввод.

question

Prompt to user *(accepts: series)*

Эта функция обеспечивает общую операцию чтобы запросить пользователя: PRIN, сопровождаемый INPUT.

```
print ask "Your name, please? "
```

```
Your name, please? tester
```

at

series index

Возвращает последовательность по указанному индексу.

series

(accepts: series)

index

Can be positive, negative, or zero. *(accepts: number)*

Обеспечивает универсальную функцию для того, чтобы индексировать в последовательность при возвращении в новом индексном пункте. Обратите внимание, что операция - относительно текущей позиции в пределах последовательности.

```
words: [grand grape great good]
```

```
print first at words 2
```

```
remove at words 2
```

```
insert at words 3 [super]
```

```
probe words
```

```
grape
```

```
[grand great super good]
```

back

series

Возвращает последовательность в его предыдущей позиции.

series

(accepts: series port)

```
str: tail "time"
```

```
until [
```

```
  str: back str
```

```
  print str
```

```
  head? str
```

```
]
```

```
e
```

```
me
```

```

ime
time
blk: tail [1 2 3 4]
until [
  blk: back blk
  print first blk
  head? blk
]
4
3
2
1

```

binary!

Когда снабжено как параметр к функции MAKE, это определяет тип значения, которое будет создано. Когда обеспечено в пределах спецификации параметра функции, просит интерпретатор проверить, что значение параметра имеет указанный тип, когда функция вычислена.

```

probe make binary! "1234"
#{31323334}

```

binary?

value

Возвращает TRUE для двоичного значения.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```

print binary? #{13FF Acd0}
true

```

bind

words known-word

Связывает слова с указанным контекстом.

words

A block of words or single word. *(accepts: block word)*

known-word

A sample word from the target context. *(accepts: word)*

/copy

Deep copies block before binding it.

Связывает значение со словами в блоке. То есть это дает словам контекст, в котором они могут интерпретироваться. Это позволяет блокам быть обмененными между различными контекстами, который разрешает их словам быть вычисленными. Например функция может обработать слова в глобальной базе данных. Второй параметр, чтобы СВЯЗАТЬ - слово от контекста, в котором блок должен быть связан. Обычно, это - слово от имеющегося контекста (например один из функциональных параметров), но это может быть слово от любого контекста в пределах системы.

```

settings: [start + duration]
schedule: function [start] [duration] [
  duration: 1:00
  do bind settings 'start
]
print schedule 10:30
11:30

```

bitset!

Когда снабжено как параметр к функции MAKE, это определяет тип значения, которое будет создано. Когда обеспечено в пределах спецификации параметра функции, просит интерпретатор проверить, что значение параметра имеет указанный тип, когда функция вычислена.

```

test-bits: make bitset! "123"
print find test-bits "2"
true

```

bitset?

value

Возвращает TRUE для значения bitset.

value *(accepts: any-type)*
Возвращает FALSE для всех других значений.
`print bitset? charset "abc"`
true
`print bitset? 123`
false

block!

Когда снабжено как параметр к функции MAKE, это определяет тип значения, которое будет создано. Когда обеспечено в пределах спецификации параметра функции, просит интерпретатор проверить, что значение параметра имеет указанный тип, когда функция вычислена.

```
block: make block! 100
repeat n 10 [insert block random n * 10]
print block
15 14 72 53 52 17 36 20 12 5
```

block?

value

Возвращает TRUE для значения **block**.

value *(accepts: any-type)*
Возвращает FALSE для всех других значений.
`print block? "1 2 3"`
false
`print block? [1 2 3]`
true
`data: load "1 2 3"`
`if block? data [print data]`
1 2 3

break

Остановка вычислений из циклов, while, until, repeat, foreach, и т. д.

/return Forces the loop function to return a value.

value *(accepts: any-type)*
Эта функция может быть помещена где-нибудь в пределах повторяемого блока, даже в пределах блока sub или функции.

```
m: 0
repeat n 5 [
  print n
  if n > 3 [ break ]
  m: n
]
1
2
3
4
```

catch

block

Перехват хода из блока, и возврат значения.

block Block to evaluate *(accepts: block)*

/name Catches a named throw.

name One or more names *(accepts: word block)*

```
print catch [
  if exists? %reboldoc.r [throw "Doc found"]
  "Doc not found"
]
Doc found
```

change

series value

Заменяет значение в последовательности и возвращает последовательность после замены.

series Series at point to change (*accepts: series port*)

value The new value (*accepts: any value*)

/part Limits to a given length or position.

range (*accepts: number series port*)

/only Changes a series as a series.

/dup Duplicates the change a specified number of times.

count (*accepts: number*)

Если второй параметр - простое значение, это заменит значение в текущей позиции в первом ряду. Если второй параметр - последовательность, совместимый с первым (блочный или datatype на основе строки), все его значения заменят таковые из первого параметра или последовательности.

```
title: copy "how it REBOL"
change title "N"
print title
Now it REBOL
change find title "t" "s"
print title
Now is REBOL
blk: copy ["now" 12:34 "REBOL"]
change next blk "is"
print blk
now is REBOL
```

change-dir

dir

Заменяет путь к основной рабочей папке.

dir new directory path (*accepts: file*)

Заменяет значение system/script/path. Это значение используется для всех связанных с файлом операций. Любой путь к файлу, который не начинается с наклонной черты вправо (/), будет относительно текущего заданного по умолчанию пути в system/script/path. Когда файл сценария выполнен, путь будет автоматически установлен на каталог, содержащий путь.

```
current: what-dir
change-dir %../
change-dir current
```

char!

Когда снабжено как параметр к функции MAKE, это определяет тип значения, которое будет создано. Когда обеспечено в пределах спецификации параметра функции, просит интерпретатор проверить, что значение параметра имеет указанный тип, когда функция вычислена.

```
char: #"A"
print type? char
char
print char
A
print head insert "BCD" char
ABCD
```

char?

value

Возвращает TRUE для значения char.

value (*accepts: any-type*)

Возвращает FALSE для всех других значений.

```
print char? #"1"
true
print char? 1
false
```

charset

chars

Делает `bitset` символов для функции синтаксического анализа.

`chars`

(*accepts: string block*)

```
digits: charset "0123456789"
area-code: ["(" 3 digits ")"]
phone-num: [3 digits "-" 4 digits]
print parse "(707)467-8000" [[area-code | none] phone-num]
true
```

checksum

data

Возвращает CRC или TCP как 16-разрядную контрольную сумму.

`data`

Data to checksum (*accepts: any-string*)

`/tcp`

Returns an Internet TCP 16-bit checksum.

Вообще, контрольная сумма - номер, который сопровождает данные, чтобы проверить, что данные не изменились (не имели ошибок).

```
print checksum "now is the dawning"
9076108
print checksum "now is the tawning"
4193466
```

clear

series

Удаляет все значения из текущего индекса.

`series`

(*accepts: series port*)

```
str: copy "with all things considered"
clear skip str 8
print str
with all
```

close

port

Закрывает открытое подключение порта.

`port`

(*accepts: port*)

```
; rebol-port: open http://www.REBOL.com
; if find rebol-port "REBOL" ["You did it!"]
; close rebol-port
```

comment

value

Игнорирует значение параметра и не возвращает ничего.

`value`

A string, block, or any other value (*accepts: any value*)

Эта функция может использоваться, чтобы добавить комментарии к сценарию или удалять блок из вычислений. Обратите внимание, что эта функция только эффективна в вычисленном коде и не имеет никакого эффекта в блоках данных. То есть в пределах комментариев блока данных появится как данные. Во многих случаях, используя COMMENT не обязателен. Размещение фигурных скобок вокруг кода предотвратит от вычислений (пока это не принадлежит другому выражению, типа IF или WHILE).

```
comment "This is a comment."
comment [
  print "Don't evaluate this."
]
```

complement

value

Возвращает поразрядное значение дополнения.

`value`

(*accepts: logic char number tuple bitset*)

Используемый для побитовой маскировки целых чисел и инвертирующий bitsets.

```
print complement 0
-1
```

compress

data

Сжимает строковую последовательность и возвращает.

`data`

Data to compress (*accepts: any-string*)

Как со всеми сжатыми файлами, сохраните распаковываемую копию оригинального файла данных как резервную копию.

```
print compress "now is the dawning"  
#{789CCB2F57C82C5628C9485548492CCFCBCC4B07003EB606BA1200000}
```

copy

value

Возвращает копию значения.

value Usually a series (*accepts: series port bitset*)

/part Limits to a given length or position.

range (*accepts: number series port*)

/deep Also copies series values within the block.

```
str: copy "with all things considered"  
print str  
with all things considered  
strings: copy ["how" "flies"]  
print strings  
how flies  
insert next strings "time"  
print strings  
how time flies
```

cosine

value

Возвращает тригонометрический косинус в степени.

value (*accepts: number*)

/radians Value is specified in radians.

```
print cosine 90  
0  
print (cosine 45) = (sine 45)  
true
```

datatype!

Когда снабжено как параметр к функции MAKE, это определяет тип значения, которое будет создано. Когда обеспечено в пределах спецификации параметра функции, просит интерпретатор проверить, что значение параметра имеет указанный тип, когда функция вычислена.

datatype?

value

Возвращает TRUE для значения datatype.

value (*accepts: any-type*)

Возвращает FALSE для всех других значений.

```
print datatype? integer!  
true
```

date!

Когда снабжено как параметр к функции MAKE, это определяет тип значения, которое будет создано. Когда обеспечено в пределах спецификации параметра функции, просит интерпретатор проверить, что значение параметра имеет указанный тип, когда функция вычислена.

```
when: make date! [30 6 1999]  
print when  
30-Jun-1999  
print date? "ABC"  
false
```

date?

value

Возвращает TRUE для значения date.

value (*accepts: any-type*)

Возвращает FALSE для всех других значений.

```
print date? 1/3/69  
true  
print date? 12:39  
false
```

decimal!

Когда снабжено как параметр к функции MAKE, это определяет тип значения, которое будет создано. Когда обеспечено в пределах спецификации параметра функции, просит интерпретатор проверить, что значение параметра имеет указанный тип, когда функция вычислена.

```
value: make decimal! "3.1415"  
print value  
3.1415
```

decimal?

value

Возвращает TRUE для значения **decimal**.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print decimal? 1.2  
true  
print decimal? 1  
false
```

decompress

data

Разворачивает двичную последовательность назад в строку.

data

Data to decompress *(accepts: binary)*

```
print decompress #{789C2B492D2E0100045D01C104000000}  
test
```

delete

target

Удаляет указанный файл (ы).

target

the file to delete *(accepts: file url)*

/any

allow wild cards

Удаляет файл или объект URL, который определен. Если файл или URL обращаются к пустому каталогу, то это будет удалено.

```
write %delete-test.r "This file is no longer needed."  
delete %delete-test.r
```

detab

string

Преобразовывает пробелы в строку. (tab size 4)

string

(accepts: any-string)

/size

Specifies the number of spaces per tab.

number

(accepts: integer)

В REBOL размер позиции табуляции значения по умолчанию - четыре пробела. DETAB удалит позиции табуляции из полной строки даже вне первого пробела.

```
text: " lots of tabs here"  
print detab text  
lots of tabs here
```

dir?

target

Возвращает TRUE если файл или URL каталог.

target

(accepts: file url)

Возвращает FALSE для всех других значений

```
print dir? %reboldoc.r  
false
```

disarm

error

Возвращает значение ошибки как объект.

error

(accepts: error)

```
probe disarm try [1 + "x"]
make object! [
  code: 304
  type: script
  id: expect-set
  arg1: string!
  arg2: integer!
  arg3: none
  near: [1 + "x"]
  where: none
]
```

divide

value1 value2

Возвращает первое значение, разделенное на второе.

value1

(accepts: char number money time tuple)

value2

(accepts: char number money time tuple)

Если второе значение является нулевым, произойдет ошибка. При делении значений разного datatypes, они должны быть совместимы.

```
print divide 123.1 12
10.258333333333333
print divide 10:00 4
2:30
```

do

value

Оценивает блок, файл, URL, функцию, слово, или любое другое значение.

value

Normally a file name, URL, or block *(accepts: any value)*

/args

If value is a script, this will set its system/script/args

arg

Args passed to a script. Normally a string. *(accepts: any value)*

Параметр к этой функции - типично блок, имя файла, URL, или строка. Если это блок, значения в блоке будут оценены. Если значение - STRING, это будет оттранслировано тогда оцененным. Если значение - FILE или URL, это будет загружено и оценено.

Результат оценки будет возвращен.

```
do "repeat n 3 [print n]"
print do "1 + 2"
1
2
3
3
do [ print "doing" ]
print do [1 + 2]
doing
3
blk: [
  [print "test"]
  [loop 3 [print "loop"]]
]
do second blk
loop
loop
loop
do first blk
test
```

echo

target

Консольные копии вывода к файлу.

target

(accepts: file none)

```
echo %helloworld.txt
print "Hello World!"
echo none
Hello World!
```

either*condition true-block false-block*

Если условие TRUE, вычисляет первый блок, если иначе то второй.

condition (*accepts: any value*)

true-block (*accepts: block*)

false-block (*accepts: block*)

```

grade: 72
either grade > 60 [
    print "Passing Grade!"
][
    print "Failing Grade!"
]
Passing Grade!
print either grade > 60 ["Passing"]["Failing"]

Passing

```

email!

Когда снабжено как параметр к функции MAKE, это определяет тип значения, которое будет создано. Когда обеспечено в пределах спецификации параметра функции, просит интерпретатор проверить, что значение параметра имеет указанный тип, когда функция вычислена.

```

print make email! ["user" "domain.com"]
user@domain.com

```

email?*value*

Возвращает TRUE для значения email.

value (*accepts: any-type*)

Возвращает FALSE для всех других значений.

```

print email? info@rebol.com
true
print email? http://www.REBOL.com
false

```

empty?*series*

Возвращает TRUE если к последовательности is at its tail.

series (*accepts: series port*)

```

print empty? []
true
print empty? [1]
false

```

entab*string*

Конвертирует пробелы в строке к табуляции. (tab size 4)

string (*accepts: any-string*)

/size Specifies the number of spaces per tab.

number (*accepts: integer*)

```

text: " lots of tabs here"
print detab text
lots of tabs here

```

equal?*value value*

Возвращает TRUE если значения равны.

value (*accepts: any value*)

value (*accepts: any value*)

Строки равны когда они индетичны или отичаются только регистром.

```

print equal? 123 123
true

```

```

print equal? "abc" "abc"
true
print equal? [1 2 3] [1 2 4]
false
print equal? 15-June-1998 15-June-1999
false
print equal? 1.2.3.4 1.2.3.0
false
print equal? 1:23 1:23
true

```

error!

Когда снабжено как параметр к функции MAKE, это определяет тип значения, которое будет создано. Когда обеспечено в пределах спецификации параметра функции, просит интерпретатор проверить, что значение параметра имеет указанный тип, когда функция вычислена.

```

;throw make error! "User Error!"

```

error?

value

Возвращает TRUE для значения error.

value

(accepts: any-type)

Возвращает FALSE для всех других значений. Это полезно для того, чтобы определить, возвратила ли функция TRY ошибку

```

if error? try [1 + "x"] [

```

```

print "Did not work."]

```

```

Did not work.

```

even?

Number

Возвращает TRUE если номер является четным.

number

(accepts: char number date money time)

```

print even? 100

```

```

true

```

```

print even? 7

```

```

false

```

exists?

target

Определяет, существуют ли файл или URL.

target

(accepts: file url)

Возвращает FALSE для всех других значений.

```

print exists? %reboldoc.r

```

```

true

```

```

print exists? %doc.r

```

```

false

```

exit

Выходит из функции, не возвращая никакого значение.

```

test-str: make function! [str] [

```

```

    if not string? str [ exit ]

```

```

    print str

```

```

]

```

```

test-str 10

```

```

test-str "right"

```

```

right

```

exp

power

Повышает E (натуральное число) к указанной степени.

power

(accepts: number)

```

print exp 3

```

false**Логика**

```
print false
false
print not true
false
print 1 > 2
false
```

fifth**series**

Возвращает пятое значение последовательности.

series

(accepts: series date port tuple)

```
print fifth "REBOL"
L
print fifth [11 22 33 44 55 66]
55
```

file!

Когда снабжено как параметр к функции MAKE, это определяет тип значения, которое будет создано. Когда обеспечено в пределах спецификации параметра функции, просит интерпретатор проверить, что значение параметра имеет указанный тип, когда функция вычислена.

```
print make file! "image.jpg"
image.jpg
```

file?**value**

Возвращает TRUE для значения file.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print file? %reboldoc.r
true
print file? "REBOL"
false
```

find**series value**

Находит значение в последовательности и возвращает последовательность в начале этого.

series

(accepts: series port bitset)

value

(accepts: any-type)

/part

Limits the search to a given length or position.

range

(accepts: number series port)

/only

Treats a series value as a single value.

/case

Characters are case-sensitive.

/any

Enables the * and ? wildcards.

/with

Allows custom wildcards.

wild

Specifies alternates for * and ? *(accepts: string)*

/match

Performs comparison and returns the tail of the match.

/tail

Returns the end of the string.

Обратите внимание: усовершенствование применяется к блочным значениям и игнорируется для строк.

```
print find "here and now" "and"
and now
print find [12:30 123 c@d.com] 123
12:30 123 c@d.com
print find [1 2 3 4] 5
```

```

none
print find/match "here and now" "here"
  and now
print find/match "now and here" "here"
none

```

first

series

Возвращает первое значение последовательности.

series (*accepts: series money date object port time tuple any-function*)

```

print first "REBOL"
R
print first [11 22 33 44 55 66]
11
print first 1:30
1
print first 199.4.80.1
199
print first 12:34:56.78
12

```

for

word start end bump body

Повторяет блок по диапазону значений.

'word Variable to hold current value (*accepts: word*)
start Starting value (*accepts: number series money time date char*)
end Ending value (*accepts: number series money time date char*)
bump Amount to skip each time (*accepts: number money time char*)
body Block to evaluate (*accepts: block*)

```

for num 0 30 10 [ print num ]
0
10
20
30
for num 4 -37 -15 [ print num ]
4
-11
-26

```

forall

word body

Вычисляет блок для каждого значения в последовательности.

'word Word set to each position in series and changed as a result (*accepts: word*)
body Block to evaluate each time (*accepts: block*)

До вычисления, параметр WORD должен быть установлен на желательную исходную позицию в пределах ряда. Для каждой оценки блока, слово будет продвинуто к следующей позиции в пределах ряда.

```

cities: ["Eureka" "Ukiah" "Santa Rosa" "Mendocino"]
forall cities [print first cities]
Eureka
Ukiah
Santa Rosa
Mendocino
chars: "abcdef"
forall chars [print first chars]
a
b
c
d
e
f

```

foreach

word data body

Вычисляет блок для каждого значения (й) в последовательности (ряде).

'word Word or block of words to set each time (will be local) (*accepts: get-word word block*)

data The series to traverse (*accepts: series*)
body Block to evaluate each time (*accepts: block*)

Для каждой оценки блока, слово будет установлено на первое значение в ряде. Параметр WORDS может быть единственным(отдельным) словом или блоком слов. Идентичный REPEAT.

```
cities: ["Eureka" "Ukiah" "Santa Rosa" "Mendocino"]
foreach city cities [print city]
Eureka
Ukiah
Santa Rosa
Mendocino
chars: "abcdef"
foreach char chars [print char]
a
b
c
d
e
f
```

form *value*

Конвертирует значение в строку.

value The value to form (*accepts: any value*)

```
print form 10:30
10:30
print form %image.jpg
image.jpg
str: form [123 "-string-" now]
print str
123 -string- now
print reform [123 "-string-" now]
123 -string- 11-May-1999/8:47:30-7:00
```

forskip *word skip-num body*

Вычисляет блок для периодических значений в ряде.

'word Word set to each position in series and changed as a result (*accepts: word*)
skip-num Number of values to skip each time (*accepts: integer*)
body Block to evaluate each time (*accepts: block*)

До оценки, слово должно быть установлено на желательную исходную позицию в пределах ряда. После каждой оценки блока, позиция слова в ряде изменена, пропуская номер значений, указанных вторым параметром (см. функцию SKIP).

```
areacodes: [
  "Ukiah"          707
  "San Francisco" 415
  "Sacramento"    916
]
forskip areacodes 2 [
  print [ first areacodes "area code is" second areacodes]
]
Ukiah area code is 707
San Francisco area code is 415
Sacramento area code is 916
```

found? *value*

Возвращает TRUE если значение NONE.

value (*accepts: any value*)
Returns FALSE for all other values.

```
if found? find carl@rebol.com ".com" [print "found it"]
found it
```

fourth *series*

Возвращает четвертое значение ряда.

series (accepts: series date port tuple)

```
print fourth "REBOL"
0
print fourth [11 22 33 44 55 66]
44
print fourth 199.4.80.1
1
```

func *spec body*

Определяет функцию пользователя.

spec Help string (opt) followed by arg words (and opt type and string) (accepts: block)

body The body block of the function (accepts: block)

```
sum: func [a b] [a + b]
print sum 123 321
444
print function? 'sum
false
print-word: func ['word] [print form word]
print-word testing
testing
ref: func [a /plus b][
  if plus [a: a + b]
  a
]
print ref 123
print ref/plus 10 20
123
30
```

function *spec vars body*

Определяет функцию пользователя с локальными словами.

spec Optional help info followed by arg words (and optional type and string) (accepts: block)

vars List of words that are local to the function (accepts: block)

body The body block of the function (accepts: block)

FUNCTION идентична FUNC, но включает блок для того, чтобы определить свои параметры к функции.

```
average: function [block] [total] [
  total: 0
  foreach number block [total: number + total]
  total: total / length? block
]
print average [1 10 12.34]
7.78
```

function!

```
test: make function! [str] [
  print str
]
test "simple"
simple
print do (make function! [n] [n * 123]) 10
1230
```

function? *value*

Возвращает TRUE для значения function.

value (accepts: any-type)

Возвращает FALSE для всех других значений.

```
print function? :func
true
```

get *word*

Получает значение слова.

word Word to get (*accepts: any-word*)

/any Allows any type of value, even unset.

Параметр GET должен быть словом, параметр должен в кавычках или извлечен из блока. Чтобы получить значение слова, постоянно находящегося в блоке, используйте функцию IN.

```
foo: true
bar: false
baz: false
words: [foo bar baz]
if get first words [print "it's true!"]
print get in system/console 'prompt'
it's true!
>>
```

get-word!

```
probe make get-word! "test"
:test
```

get-word?

value

Возвращает TRUE для значения get-word.

value (*accepts: any-type*)

Возвращает FALSE для всех других значений.

```
print get-word? first [:foo bar]
true
```

greater?

value value

Возвращает TRUE, если первое значение больше чем второе.

value (*accepts: any value*)

value (*accepts: any value*)

```
print greater? "abc" "abb"
true
print greater? 16-June-1999 15-June-1999
true
print greater? 4.3.2.1 1.2.3.4
true
print greater? 11:00 12:00
false
```

greater-or-equal?

value value

Возвращения TRUE, если первое значение больше или равно второму значению.

value (*accepts: any value*)

value (*accepts: any value*)

```
print greater-or-equal? "abc" "abb"
true
print greater-or-equal? 16-June-1999 15-June-1999
true
print greater-or-equal? 1.2.3.4 4.3.2.1
false
print greater-or-equal? 1:00 11:00
false
```

halt

Остановка вычисления и возврат к входящей подсказки.

```
name: 10
if not string? name [
  print "not a string"
; halt
```

```
]
not a string
```

hash!

```
blk: ["Doug" "Mary" "Kas" "Jodi" "Patrick" "Polly" "Eli"]
hash: make hash! blk
find hash "Polly"
```

hash?

value

Возвращает TRUE для значения hash.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
test-hash: make hash! [1 2 3 4]
if hash? test-hash [print "It's a hash."]
It's a hash.
```

head

series

Возвращает последовательность в головную часть.

series

(accepts: series port)

```
str: "with all things"
here: find str "all"
print here
all things
print head here
with all things
```

head?

series

Возвращает TRUE если последовательность в головной части.

series

(accepts: series port)

```
cds: tail [
  "Rockin' REBOLs"
  "Carl's Addiction"
  "Jumpin' Jim"
]
until [
  cds: back cds
  print first cds
  head? cds
]
Jumpin' Jim
Carl's Addiction
Rockin' REBOLs
```

help

word

Помощь.

'word

(accepts: any-type)

help quit

Stops evaluation and exits the interpreter.

if

condition block

Если условие TRUE, block вычисляется.

condition

(accepts: any value)

block

(accepts: block)

```
age: 4
if age > 3 [print "wine is ready"]
wine is ready
```

in

object word

Возвращает слово в контексте объекта.

object (accepts: object)
word (accepts: word)

```
set-console: func ['word value] [  
    set in system/console word value  
]  
set-console prompt "="  
set-console prompt ">>"
```

index?

series

Возвращает индексный номер текущей позиции в ряде.

series (accepts: series port)

```
str: "with all things considered"  
print index? str  
1  
print index? find str "things"  
10  
blk: [264 "Rebol Dr." "Calpella" CA 95418]  
print index? find blk 95418  
5
```

info?

target

Возвращает информацию о файле или url.

target (accepts: file url)

```
info: info? %reboldoc.r  
print info/size  
print info/date  
98384  
11-May-1999/8:47:02-7:00
```

input

Возвращает строку из стандартного устройства ввода данных.

Возвращает строку из стандартного устройства ввода данных (обычно клавиатура, но может также быть файл или интерактивная оболочка)..

```
prin "Enter a name: "  
name: input  
print [ name "is" length? name "characters long"]  
Enter a name: tester is 6 characters long
```

input?

INPUT может использоваться, чтобы определить, ждет ли ввод, чтобы читаться с стандартного устройства ввода данных.

```
print input?  
true  
if input? [name: input]
```

insert

series value

Вставляет значение в ряд и возвращает ряд после вставки.

series Series at point to insert (accepts: series port bitset)
value The value to insert (accepts: any-type)
/part Limits to a given length or position.
range (accepts: number series port)
/only Inserts a series as a series.
/dup Duplicates the insert a specified number of times.
count (accepts: number)

```
blk: [1 2]  
insert next blk '+  
print blk  
3
```

```
blk: [10:30 "Test"]
insert next blk [1 2 3]
print blk
10:30 1 2 3 Test
str: "with things considered"
insert (find str "things") "all "
print str
with all things considered
```

integer!

Целое число.

```
print make integer! "12345"
12345
```

integer?

value

Возвращает TRUE для значения integer.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print integer? -1234
true
print integer? "string"
false
```

issue!

```
print make issue! "1234-56-7890"
1234-56-7890
```

issue?

value

Возвращает TRUE для значения issue.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print issue? #1234-5678-9012
true
print issue? $12.56
false
```

join

value rest

Связывает значения.

value

Base value (accepts: any value)

rest

Value or block of values (accepts: any value)

Создает новое значение. Первый параметр определит datatype объединения. Например, STRING, FILE, URL, BLOCK может использовать JOIN. Если второй параметр - блок, это будет вычисленно, все его значениям будут присоединяться к строке.

```
probe join "super" "duper"
"superduper"
probe join %file ".txt"
%file.txt
probe join "t" ["e" "s" "t"]
"test"
```

last

series

Возвращает последнее значение ряда.

series

(accepts: series port tuple)

```
print last "REBOL"
L
print last [11 22 33 44 55 66]
66
```

length?

series

Возвращает длину ряда от текущей позиции.

series *(accepts: series port tuple)*
`print length? "REBOL"`
5
`print length? [1 2 3 [4 5]]`
4

lesser? *value value*

Возвращает TRUE, если первое значение - меньше чем второе.

value *(accepts: any value)*

value *(accepts: any value)*

```
print lesser? "abc" "abcd"  
true  
print lesser? 15-June-1999 14-June-1999  
false  
print lesser? 1.2.3.4 4.3.2.1  
true  
print lesser? 1:30 2:00  
true
```

lesser-or-equal? *value value*

Возвращения TRUE, если первое значение меньше или равно второму значению.

value *(accepts: any value)*

value *(accepts: any value)*

```
print lesser-or-equal? "abc" "abd"  
true  
print lesser-or-equal? 14-June-1999 15-June-1999  
true  
print lesser-or-equal? 4.3.2.1 1.2.3.4  
false  
print lesser-or-equal? 1:23 10:00  
true
```

license

О лицензионном соглашении REBOL/core.

```
license
```

list!

```
blk: ["Carla" "Brian" "Karen" "Adam" "Jeremy"]  
list: make list! blk  
remove list  
append list "Susan"
```

list? *value*

Возвращает TRUE для значения list.

value *(accepts: any-type)*

Возвращает FALSE для всех других значений.

```
test-list: make list! [1 2 3 4]  
if list? test-list [print "It's a list."]  
It's a list.
```

list-dir *dir*

Печатает многоколоночную сортированную распечатку каталога.

dir Directory to list or nothing *(accepts: any-type)*

```
list-dir  
do-demo.r          feedback.r          helloworld.txt     nntp.r  
rebdoc.r           REBOL.exe          rebol.r            rebol0316.zip  
reboldoc.r        user.r
```

lit-word!

```
probe make lit-word! "test"
'test
```

lit-word?

value

Возвращает TRUE для значения lit-word.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print lit-word? first ['foo bar]
true
```

load

source

Загружает файл, URL, или строку. Связывает слова с глобальным контекстом.

source

(accepts: file url string any-block)

/header

Includes REBOL header object if present.

```
data: load "11 22 33 44"
print third data
33
```

log-10

value

Возвращает base-10 логарифм.

value

(accepts: number)

```
print log-10 1234
3.09131515969722
```

log-2

value

Возвращает base-2 логарифм.

value

(accepts: number)

```
print log-2 1234
10.2691266791494
```

log-e

value

Возвращает base-E (натуральный номер) логарифм.

value

(accepts: number)

```
print log-e 1234
7.11801620446533
print exp log-e 1234
1234
```

logic!

```
print make logic! 1
true
```

logic?

value

Возвращает TRUE для значения logic.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print logic? true
true
print logic? 123
false
```

lowercase

string

Строка конвертированных символов в нижний регистр.

string

(accepts: any-string)

/part

Limits to a given length or position.

range

(accepts: integer any-string)

```
print lowercase "ABCDEF"
abcdef
print lowercase/part "ABCDEF" 3
abcDEF
```

loop

count block

Вычисляет блок указанное число раз. Цикл.

count Number of repetitions (*accepts: integer*)

block Block to evaluate (*accepts: block*)

```
loop 40 [prin "***"]
print ""
*****
```

make

type spec

Конструирование и возврат нового значения.

type The datatype or example value. (*accepts: any-type*)

spec The attributes of the new value. (*accepts: any-type*)

```
cash: make money! 1234.56
print cash
time: make time! [10 30 40]
print time
$1234.56
10:30:40
```

make-dir

dir

Создает новый какталог.

dir path to new directory (*accepts: file url*)

```
make-dir %"New Directory"
delete %"New Directory/"
```

maximum

value1 value2

Возвращает больше из двух значений.

value1 (*accepts: char number money date time tuple series*)

value2 (*accepts: char number money date time tuple series*)

```
print maximum "abc" "abd"
abd
print maximum 12:00 11:00
12:00
```

minimum

value1 value2

Возвращает меньший из двух значений.

value1 (*accepts: char number money date time tuple series*)

value2 (*accepts: char number money date time tuple series*)

```
print maximum "abc" "abd"
abd
print maximum 12:00 11:00
12:00
```

modified?

target

Возвращает дату последнего изменения файла или URL.

target (*accepts: file url*)

```
print modified? %reboldoc.r
11-May-1999/8:47:02-7:00
```

mold

value

Конвертирует значение к читаемой в REBOL строке.

value The value to mold (*accepts: any value*)

```
print mold 10:30
```

```
10:30
print mold %image.jpg
%image.jpg
print mold [1 2 3]
[1 2 3]
```

money!

Деньги.

```
print make money! "123"
$123.00
```

money?

value

Возвращает TRUE для значения **money**.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print money? $123
true
print money? 123.45
false
```

multiply

value1 value2

Возвращает первое значение, умноженное на второе.

value1

(accepts: char number money time tuple)

value2

(accepts: char number money time tuple)

```
print multiply 123 10
1230
print multiply 3:20 3
10:00
print multiply 0:01 60
1:00
```

native!

native?

value

Возвращает TRUE для значения **native**.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print native? :native?
false
print native? "Vichy"
false
```

negative?

number

Возвращает TRUE если число **negative**.

number

(accepts: char number money time)

Возвращает FALSE для всех других значений.

```
print negative? -50
true
print negative? 50
false
```

negate

number

Изменяет признак числа.

number

(accepts: number money time bitset)

```
print negate 123
-123
print negate 123.45
-123.45
```

newline

Возвращает строку, которая, когда напечатано, начинает новую строку.

```
print "start"
print newline
print "end"
start
end
```

next

series

Возвращает ряд в его следующей позиции.

series

(*accepts: series port*)

```
str: "REBOL"
loop length? str [
  print str
  str: next str
]
REBOL
EBOL
BOL
OL
L
blk: ["red" "green" "blue"]
loop length? blk [
  print blk
  blk: next blk
]
red green blue
red green blue
red green blue
```

no

Другое определение для логического FALSE.

```
print no
false
print not no
true
```

none

Единственное значение, которое не означает ни чего.

```
print none
none
```

none!

none?

value

Возвращает TRUE для значения none.

value

(*accepts: any-type*)

Возвращает FALSE для всех других значений.

```
print none? NONE
true
print none? pick "abc" 4
true
print none? find "abc" "d"
true
```

not

value

Возвращает логическое дополнение.

value

(Only FALSE and NONE return TRUE) (*accepts: any value*)

Функция возвращает TRUE, если значение является FALSE, и FALSE, если это TRUE. К

поразрядному дополнению INTEGER, используйте функцию COMPLEMENT.

```
print not true
false
print not (10 = 1)
true
```

not-equal?

value value

Возвращает TRUE, если значения не равны.

value (*accepts: any value*)

value (*accepts: any value*)

```
print not-equal? "abc" "abcd"
true
print not-equal? [1 2 4] [1 2 3]
true
print not-equal? 12-sep-98 10:30
true
```

now

Возвращает текущее локальное время и дату.

/year Returns the year only.

/month Returns the month only.

/day Returns the day of the month only.

/time Returns the time only.

/zone Returns the time zone offset from GMT only.

/date Returns date only.

/weekday Returns day of the week as integer (Sunday is day 7).

```
print now
11-May-1999/8:47:31-7:00
print now/date
11-May-1999
print now/time
8:47:31
print now/zone
-7:00
print now/weekday
2
```

number!

```
times: func [a [number!] b [number!]] [a * b]
print times 10 1.5
15
```

number?

value

Возвращает TRUE для значения number.

value (*accepts: any-type*)

Возвращает FALSE для всех других значений.

```
print number? 1234
true
print number? 12.34
true
print number? "1234"
false
```

object!

```
test: make object! [
  name: "alpha"
  size: 12
  type: 'rebol
```

```
]
print test/name
alpha
```

object?

value

Возвращает TRUE для значения object.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print object? REBOL
true
```

odd?

number

Возвращает TRUE if если число odd (нечетное).

number

(accepts: char number date money time)

```
print odd? 3
true
print odd? 100
false
print odd? 0
false
```

op!

op?

value

Возвращает TRUE для значения op.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print op? :and
true
```

open

spec

Открывает новое подключение порта.

spec

(accepts: file url port object block)

/binary

Preserves contents exactly.

/string

Translates all line terminators.

/direct

Opens the port without buffering.

/new

Creates a file. (Need not already exist.)

/read

Read only. Disables write operations.

/write

Write only. Disables read operations.

/wait

Waits for data.

/lines

Handles data as lines.

/with

Specifies alternate line termination.

end-of-line

(accepts: char string)

/allow

Specifies the protection attributes when created.

access

(accepts: block)

/mode

Block of above refinements.

args

(accepts: block)

/custom

Allows special refinements.

params

(accepts: block)

```
autos: open/new %autos.txt
insert autos "Ford"
insert tail autos " Chevy"
close autos
```

```
print read %autos.txt
Ford Chevy
```

or

value1 value2

Возвращает первое значение ORed со вторым.

value1 (*accepts: logic char number tuple*)

value2 (*accepts: logic char number tuple*)

Инфиксный оператор. Для LOGIC значений, оба должны быть FALSE, чтобы вернуть FALSE; иначе возвращается TRUE. Для целых чисел, отдельно затрагивают каждый бит. Поскольку это - инфиксный оператор, OR должен быть между двумя значениями.

```
print 122 or 1
123
print true or false
true
print (10 > 20) or (20 < 100)
true
print 1.2.3.4 or 255.255.255.0
255.255.255.4
```

paren!

```
print mold make paren! [1 + 2]
(1 + 2)
```

paren?

value

Возвращает TRUE для значения paren.

value (*accepts: any-type*)

Возвращает FALSE для всех других значений. **paren** идентичен блоку, но немедленно вычисляется когда будет найден.

```
print paren? second [123 (456 + 7)]
true
print paren? last [1 2 3]
false
```

parse

input rules

Анализирует ряд согласно правилам.

input Input series to parse (*accepts: any-string*)

rules Rules to parse by (*accepts: block string none*)

/all Parses all chars including spaces.

/case Uses case-sensitive comparison.

Синтаксический анализ обеспечивает средство определения ряда символов, которые появляются в специфическом порядке. По существу, это – метод поиска и разбора в соответствии с образцами. Параметр правил снабжает блок грамматикой. Есть также простой режим синтаксического анализа, который не требует правил, но берет строку символов, чтобы использовать для того, чтобы разбить входную строку. Синтаксический анализ также работает вместе с bitsets (charset), чтобы определить группы специальных символов. Результат, возвращенный из простых синтаксических анализов - блок значений. Для синтаксических анализов на основе правила, если синтаксический анализ удачен, вернет TRUE. Все усовершенствования указывают, что все символы в пределах строки анализируются. Иначе, пробелы, позиции табуляции, newlines, и другие непечатаемые символы будут обработаны как пробелы.

```
print parse "divide on spaces" none
divide on spaces
print parse "Harry Haiku, 264 River Rd., Ukiah, 95482" ", "
Harry Haiku 264 River Rd. Ukiah 95482
page: read %reboldoc.r
```

```
parse page [thru
```

copy title to

]

print title copy title to

```
digits: charset "0123456789"  
area-code: ["(" 3 digits ")"]  
phone-num: [3 digits "-" 4 digits]  
print parse "(707)467-8000" [[area-code | none] phone-num]  
true
```

path!

path?

value

Возвращает TRUE для значения path.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print path? first [random/seed 10]  
true
```

pick

series index

Возвращает значение указанной позиции в ряде.

series

(accepts: series money date time object port tuple any-function)

index

(accepts: number logic)

Значение выбрано относительно текущей позиции в ряде (не обязательно в начале). Параметр VALUE может быть INTEGER или LOGIC. Положительное целое число позиционирует вперед, негатив позиционирует назад. Если INTEGER - вне диапазона, возвращен NONE. Если значение является LOGIC, то TRUE относится к первой позиции и FALSE к секунде (тот же самый порядок). Попытка выбирать значение вне пределов ряда не возвратит NONE.

```
str: "REBOL"  
print pick str 2  
E  
print pick 199.4.80.1 3  
80
```

poke

value index data

Возвращаемые значения после изменения его данных по данному индексу. (См. руководство)

value

(accepts: series money date time object port tuple)

index

(accepts: number logic)

data

new value *(accepts: any value)*

Похож на CHANGE, но еще и берет индекс в последовательность.

```
print poke 1.2.3.4 2 10  
1.10.3.4
```

port!

port?

value

Возвращает TRUE для значения port.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print port? port!  
false
```

positive? *number*

Возвращает TRUE если значение положительное.

number (*accepts: char number money time*)

Возвращает FALSE для всех других значений.

```
print positive? 50  
true  
print positive? -50  
false
```

power *number number*

Возвращает первое число возведенное в степень вторым числом.

number (*accepts: number*)

number (*accepts: number*)

```
print power 10 2  
100  
print power 12.3 5  
281530.56843
```

prin *a*

a (*accepts: any value*)

Никакое завершение строки не используется, так что следующее напечатанное значение появится на той же самой строке. Если значение - блок, каждое из его значений будет вычисленно а после напечатанно.

```
prin "The value is"
```

```
prin [1 + 2]
```

```
The value is3
```

```
print ""
```

print *a*

a (*accepts: any value*)

Если значение блок, каждое из его значений будет вычислено и распечатано.

```
print 1234
```

```
1234
```

```
print ["The value is" 1 + 2]
```

```
The value is 3
```

probe *value*

Печатает не вычисленное значение и возвращает то же самое значение.

value (*accepts: any value*)

PROBE очень полезен для отладки сценариев.

```
probe 10
```

```
10
```

```
probe :print
```

```
func [a][append append description reform a newline]
```

```
probe ["red" "green" "blue"]
```

```
["red" "green" "blue"]
```

query *port*

Возвращает информацию о файле и URL.

port (*accepts: file url block port*)

Его параметр - нераскрытая спецификация порта. Размер, дата и поля состояния в спецификации порта будут модифицированы с соответствующей информацией, если

запрос успешен.

quit

Остановка вычисления и выход интерпретатора.

```
noon: 12:00
if 13:30 > noon [
    print "time for lunch"
    ;quit
]
time for lunch
```

random

value

Возвращает случайное значение того же самого datatype.

value Maximum value of result (*accepts: any value*)

/seed Restart or randomize

Значение может использоваться, чтобы ограничить диапазон случайного результата. Для целых чисел случайный начинается с единицы содержащейся в значении. Чтобы инициализировать генератор случайного числа к случайному состоянию, делайте это с текущим временем.

```
loop 4 [print random 10]
1
2
7
10
lunch: ["Italian" "French" "Japanese" "American"]
print pick lunch random 4
Italian
loop 3 [print random true]
true
false
true
loop 5 [print random 1:00:00]
0:49:26
0:07:21
0:37:25
0:45:04
0:05:19
random/seed now
```

read

source

Чтения из файла, url, или спец-порта (блокируют или объект).

source (*accepts: file url object block*)

/binary Preserves contents exactly.

/string Translates all line terminators.

/direct Opens the port without buffering.

/wait Waits for data.

/lines Handles data as lines.

/part Reads a specified amount of data.

size (*accepts: number*)

/with Specifies alternate line termination.

end-of-line (*accepts: char string*)

/mode Block of above refinements.

args (*accepts: block*)

/custom Allows special refinements.

params (*accepts: block*)

Это - высокоуровневая операция порта, которая открывает порт, читает некоторых или все данные, затем закрывает порт и возвращает данные, которые читала. Это - самый

простой способ получить информацию из файла или URL.

```
save %rebol-test-file.r 12345
print length? read %rebol-test-file.r
5
```

recycle

Освобождение неиспользованной памяти.

/off

/on

/torture Temporary internal feature

Полезен для проверки сборки «мусора». По умолчанию RECYCLE/ON.

```
recycle/off
```

```
recycle/on
```

reduce

value

Вычисляет блок выражений и возвращает результат как блок.

value

(accepts: any value)

```
values: reduce [1 + 2 3 + 4]
```

```
print values
```

```
3 7
```

refinement!

refinement?

value

Возвращает TRUE для значения refinement.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print refinement? /any
```

```
true
```

```
print refinement? 'any
```

```
false
```

reform

value

Формирует уменьшенный блок и возвращает строку.

value

Value to reduce and form (accepts: any value)

Идентичный FORM но сначала уменьшает его параметр.

```
print reform ["the time is:" now]
```

```
the time is: 11-May-1999/8:47:35-7:00
```

rejoin

block

Уменьшает и присоединяет к блоку значений.

block

Values to reduce and join (accepts: block)

Подобно JOIN но вводит один параметр (который будет сначала уменьшен).

```
print rejoin ["the time is:" now]
```

```
the time is:11-May-1999/8:47:35-7:00
```

remainder

value1 value2

Возвращает остаток от первого значения, разделенного на секунду.

value1

(accepts: char number money time tuple)

value2

(accepts: char number money time tuple)

Если второе целое число является нулевым, произойдет ошибка.

```
print remainder 123 10
```

```
3
```

```
print remainder 10 10
```

```
0
```

remold*value***Формирует уменьшенный блок и возвращает строку.**

value

Value to reduce and mold (*accepts: any value*)

```
print remold ["the time is:" now]
["the time is:" 11-May-1999/8:47:35-7:00]
```

remove*series***Удаляет числовые значения и возвращает после удаления.**

series

*(accepts: series port bitset)***/part**

Removes to a given length or position.

range

(accepts: number series port)

```
str: copy "email"
remove str
print str
mail
blk: copy ["red" "green" "blue"]
remove next blk
print blk
red blue
```

rename*old new***Переименование файла.**

old

path to the old file (*accepts: file url*)

new

new name (not a path) (*accepts: file url string*)

```
rename-test: open/new %testfile
close rename-test
rename %testfile %remove
delete %remove ;to clean up
```

repeat*word value body***Вычисляет блок указанное количество раз или по последовательности.**

'word

Word to set each time (*accepts: word*)

value

Maximum number or series to traverse (*accepts: integer series*)

body

Block to evaluate each time (*accepts: block*)

```
repeat num 5 [print num]
1
2
3
4
5
```

return*value***Возвращает значение из функции.**

value

(accepts: any-type)

Выходит из текущей функции пользователя немедленно, возвращая значение как результат функции. Чтобы не возвращать никакого значения, используйте функцию EXIT.

```
fun: make function! [this-time] [
  either this-time >= 12:00 [return "after noon"]
  [return "before noon"]
]
print fun 9:00
before noon
print fun 18:00
after noon
```

same?*value value***Возвращает TRUE если значения идентичны.**

value (accepts: any value)

value (accepts: any value)

Возвращает TRUE, если значения - идентичные объекты, не только в значении.

Например, TRUE была бы возвращена, если две строки - та же самая строка (занимают то же самое местоположение в памяти). Возвращает FALSE для всех других значений.

```
a: "apple"
```

```
b: a
```

```
print same? a b
```

```
true
```

```
print same? a "apple"
```

```
true
```

save *where value*

Сохраняет значение к файлу или url.

where Where to save it. (accepts: file url)

value Value to save. (accepts: any value)

/header Save it with a header

header-data Header block or object (accepts: block object)

```
save %number 1234
```

```
print read %number
```

```
1234
```

```
print load %number
```

```
1234
```

script? *value*

Проверяет файл, url, или строки для правильного заголовка сценария. (Валидность).

value (accepts: file url string)

Если заголовок найден, строка будет возвращена. Если ничего не найдено, вернет NONE.

```
if script? %reboldoc.r [print "found the script"]
```

```
found the script
```

second *series*

Возвращает второе значение ряда.

series (accepts: series money date object port time tuple any-function)

Ошибка произойдет, если никакое значение не найдено. Используйте функцию PICK, чтобы избежать этой ошибки.

```
print second "REBOL"
```

```
E
```

```
print second [11 22 33 44 55 66]
```

```
22
```

```
print second 15-Jun-1999
```

```
6
```

```
print second 199.4.80.1
```

```
4
```

```
print second 12:34:56.78
```

```
34
```

secure *level*

Изменение уровня защиты.

'level Levels are: quit, throw, ask, read, and none. (accepts: word)

SECURE препятствует сценариям обращаться к жесткому диску или компьютерной сети.

Сценарий может только запросить, чтобы защита была уменьшена. Уровень защиты может также быть изменен в командной строке, которая выполняет REBOL. Заданный по умолчанию уровень защиты - WRITE, которая позволяет читать файлы, но не писать файлы. Предлагаются несколько уровней защиты. Самый высокий уровень - QUIT, который отрицает все операции и выходит из программы. THROW отрицает операции и обеспечивает перехват ошибки в сценарии. READ позволяет файлам читаться, но запись только с разрешения. WRITE позволяет файлам читаться и записываться. NONE не

выключает SECURE.

`secure none`

select

series value

Находит значение в ряде и возвращает значение или ряд после этого.

series *(accepts: series port)*

value *(accepts: any-type)*

/part Limits the search to a given length or position.

range *(accepts: number series port)*

/only Treats a series value as a single value.

/case Characters are case-sensitive.

/any Enables the * and ? wildcards.

/with Allows custom wildcards.

wild Specifies alternates for * and ? *(accepts: string)*

Возвращает NONE для всех других значений. Подобный функции FIND, которая возвращает начало найденного значения. Вычисление усовершенствованно для блочного параметра и игнорируется, если первый параметр - строка.

```
blk: [red 123 green 456 blue 789]
```

```
print select blk 'red
```

```
123
```

```
print select blk 456
```

```
blue
```

```
str: "with all things considered"
```

```
print select str "is "
```

```
none
```

```
space: first " "
```

```
print select str space
```

```
a
```

send

address message

Посылка сообщения адресу (или блок адресов).

address An address or block of addresses *(accepts: email block)*

message Text of message. First line is subject. *(accepts: any value)*

/only Send only one message to multiple addresses

/header Supply your own custom header

header-obj The header to use *(accepts: object)*

Первый параметр может быть адресом электронной почты, портом сообщения, или блоком таких значений. Посланное значение может быть любым REBOL datatype, но это будет преобразовано в текст до отправки. Для электронной почты, заголовок сообщения будет создан из заданного по умолчанию заголовка, если посланное значение не допустимый почтовый объект, когда это будет использоваться, чтобы создать заголовок.

```
send danny@rebol.com "Testing REBOL Function Summary"
```

```
connecting to: smtp.rebol.net
```

series!

```
test-series: func [list [series!]][print length? list]
```

```
test-series "long string"
```

```
test-series [long series]
```

```
11
```

```
2
```

series?

value

Возвращает TRUE для значения series.

value *(accepts: any-type)*

Возвращает FALSE для всех других значений.

```
print series? "string"
```

```
true
print series? [1 2 3]
true
```

set *word value*

Устанавливает слово или блок слов к указанному значению (ям).

word Word or words to set (*accepts: any-word block*)

value Value or block of values (*accepts: any-type*)

/any Allows setting words to any value.

```
set 'test 1234
print test
1234
set [a b] 123
print a
123
print b
123
set [d e] [1 2]
print d
1
print e
2
```

set-net *settings*

Сетевая установка. Все значения после значения по умолчанию являются дополнительными. ОК слов для имен сервера.

settings [email-addr default-server pop-server proxy-server proxy-port-id] (*accepts: block*)

Пример закоментирован, потому что это изменило бы текущие параметры настройки.
`;set-net [user@domain.name proxy.domain.name none mail.domain.name]`

set-path!

set-path? *value*

Возвращает TRUE для значения set-path.

value (*accepts: any-type*)

Возвращает FALSE для всех других значений.

```
if set-path? first [a/b/c: 10] [print "Path okay"]
Path okay
```

set-word!

```
probe make set-word! "test"
test:
```

set-word? *value*

Возвращает TRUE для значения set-word.

value (*accepts: any-type*)

Возвращает FALSE для всех других значений.

```
probe make set-word! "test"
test:
```

sine *value*

Возвращает тригонометрический синус в градусах.

value (*accepts: number*)

/radians Value is specified in radians.

Отношение между длиной противоположной стороны к длине гипотенузы правого

```
треугольника.  
print sine 90  
1  
print (sine 45) = (cosine 45)  
true
```

size? target

Возвращает размер из содержания URL или файла.

```
target (accepts: file url)  
print size? %reboldoc.r  
98384
```

skip series offset

Возвращает ряд вперед или назад от текущей позиции.

```
series (accepts: series port)  
offset Can be positive, negative, or zero. (accepts: number)  
str: "REBOL" print skip str 3  
OL  
blk: [11 22 33 44 55]  
print skip blk 3  
11 22 33 44 55
```

sort series

Сортирует ряд.

```
series (accepts: series port)  
/case Case sensitive sort.  
/skip Treat the series as records of fixed size.  
size Size of each record. (accepts: integer)  
/compare Use a function for ordering relation.  
comparator (accepts: function)  
blk: [34 12 934]  
print sort blk  
12 34 934
```

source word

Печатает исходный текст для слова.

```
'word (accepts: word)  
source copy  
copy: native [  
  "Returns a copy of a value."  
  value [series! port! bitset!] "Usually a series"  
  /part "Limits to a given length or position."  
  range [number! series! port!]  
  /deep "Also copies series values within the block."  
]  
source source  
source: func [  
  "Print the source code for a word"  
  'word [word!]  
]  
[  
  prin join "" [word ": "]  
  if not value? word [print "undefined" exit]  
  either any [native? get word op? get word action? get word] [  
    print ["native" mold third get word]  
  ] [print mold get word]  
]
```

square-root value

Возвращает квадратный корень числа.

```
value (accepts: number)
```

```
print square-root 2
1.4142135623731
```

strict-equal?

value value

Возвращает TRUE, если значения равны и тот же самый datatype.

value *(accepts: any value)*

value *(accepts: any value)*

```
print strict-equal? 123 123
true
```

strict-not-equal?

value value

Возвращает TRUE, если значения не равны и разные datatype.

value *(accepts: any value)*

value *(accepts: any value)*

```
print strict-not-equal? 124 123
false
print strict-not-equal? 12-sep-98 10:30
true
```

string!

```
str: make string! 20
insert str "hello"
print str
hello
```

string?

value

Возвращает TRUE для значения string.

value *(accepts: any-type)*

Возвращает FALSE для всех других значений.

```
print string? "with all things considered"
true
print string? 123
false
```

subtract

value1 value2

Возвращает второе значение, вычтенное из первого.

value1 *(accepts: char number money date time tuple)*

value2 *(accepts: char number money date time tuple)*

```
print subtract 123 1
122
print subtract 1.2.3.4 1.0.3.0
0.2.0.4
print subtract 12:00 11:00
1:00
```

symbol!

symbol?

value

Возвращает TRUE для значения symbol.

value *(accepts: any-type)*

Возвращает FALSE для всех других значений.

system

Системный объект REBOL.

```
print system/version  
2.0.0.3.1
```

tag!

```
print make tag! "test"
```

tag?

value

Возвращает TRUE для значения tag.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print tag?  
true
```

tail

series

Возвращает ряд в позиции после последнего значения.

series

(accepts: series port)

```
blk: copy [11 22 33]  
insert tail blk [44 55 66]  
print blk  
11 22 33 44 55 66
```

tail?

series

Возвращает TRUE, если ряд - в его хвосте.

series

(accepts: series port)

Эта функция - лучший способ обнаружить конец ряда при перемещении через это.

items: [oven sink stove blender]

```
while [not tail? items] [  
    print first items  
    items: next items  
]
```

```
oven  
sink  
stove  
blender  
str: "Ok"  
print tail? tail str  
true  
print tail? next next str  
true  
blk: [1 2]  
print tail? tail blk  
true  
print tail? next next blk  
true
```

tangent

value

Возвращает тригонометрический тангенс в градусах.

value

(accepts: number)

/radians

Value is specified in radians.

```
print tangent 30  
0.577350269189626
```

third

series

Возвращает третье значение ряда.

series

(accepts: series date port time tuple any-function)

```
print third "REBOL"  
B
```


a

to-date *value*

Преобразовывает в значение даты.

value Value to convert (*accepts: any value*)
`print to-date "12-April-1999"`
12-Apr-1999

to-decimal *value*

Преобразовывает в десятичное значение.

value Value to convert (*accepts: any value*)
`print to-decimal 12.3`
12.3

to-email *value*

Преобразовывает в почтовое значение.

value Value to convert (*accepts: any value*)
`address: to-email ask "What is your email address? "`
`print address`
What is your email address? tester

to-file *value*

Преобразовывает в значение файла.

value Value to convert (*accepts: any value*)
`print to-file ask "What file would you like to create? "`
What file would you like to create? tester

to-get-word *value*

Преобразовывает в значение get-word.

value Value to convert (*accepts: any value*)
`probe to-get-word "test"`
:test

to-hash *value*

Преобразовывает в значение хеш.

value Value to convert (*accepts: any value*)
`test-hash: to-hash [Reggie]`
`print test-hash`
Reggie

to-hex *value*

Конвертирует целое число к шестнадцатеричному.

value Value to be converted (*accepts: integer*)
`print to-hex 123`
0000007B

to-idade *date*

Возвращает стандартную строку даты Internet.

date (*accepts: date*)
`print to-idade now`
Tue, 11 May 1999 8:47:39 -0700

to-integer *value*

Преобразовывает в целочисленное значение.

value Value to convert (*accepts: any value*)
`to-integer 123.9`

to-issue*value***Преобразовывает в значение вывода.**

value

Value to convert (*accepts: any value*)

```
print to-issue "1234-56-7890"
1234-56-7890
```

to-list*value***Преобразовывает в значение списка.**

value

Value to convert (*accepts: any value*)

```
test-list: to-list [Bo]
print test-list
insert next test-list "Scott"
print test-list
Bo
Bo Scott
```

to-lit-word*value***Преобразовывает в значение слова литерала.**

value

Value to convert (*accepts: any value*)

```
probe to-lit-word "test"
'test'
```

to-logic*value***Преобразовывает в логическое значение.**

value

Value to convert (*accepts: any value*)

Аналог MAKE LOGIC!.

```
print to-logic 1
print to-logic 0
true
false
```

to-money*value***Преобразовывает в значение денег.**

value

Value to convert (*accepts: any value*)

```
print to-money 123.4
$123.40
```

to-none*value***Преобразовывает в нулевое значение.**

value

Value to convert (*accepts: any value*)

Аналог MAKE NONE!.

to-paren*value***Преобразовывает в значение paren.**

value

Value to convert (*accepts: any value*)

```
print to-paren "123 456"
456
```

to-refinement*value***Преобразовывает в значение усовершенствования.**

value

Value to convert (*accepts: any value*)

Аналог MAKE REFINEMENT!.

to-set-path*value***Преобразовывает в значение пути набора.**

value

Value to convert (*accepts: any value*)

```
probe to-set-path "object word"
object/word:
```

to-set-word *value*

Преобразовывает в значение набора слова.

value Value to convert (*accepts: any value*)

```
probe to-set-word "test"
test:
```

to-string *value*

Преобразовывает в значение строки.

value Value to convert (*accepts: any value*)

```
print to-string [123 456]
123456
```

to-tag *value*

Преобразовывает в значение тэга.

value Value to convert (*accepts: any value*)

```
print to-tag ";comment:"
<;comment:>
```

to-time *value*

Преобразовывает в значение времени.

value Value to convert (*accepts: any value*)

```
print to-time 75
0:01:15
```

to-tuple *value*

Преобразовывает в значение кортежа.

value Value to convert (*accepts: any value*)

```
print to-tuple [12 34 56]
12.34.56
```

to-url *value*

Преобразовывает в значение url.

value Value to convert (*accepts: any value*)

```
print to-url "info@rebol.com"
info@rebol.com
```

to-word *value*

Преобразовывает в значение слова.

value Value to convert (*accepts: any value*)

```
print to-word "test"
test
```

trace *mode*

Включает и отключает просмотр вычисления.

mode (*accepts: logic*)

/net Enable/disable network tracing.

Используется чтобы отлаживать сценарий. TRACE ON включает. TRACE OFF выключает. Чтобы ограничивать количество вывода, выборочное место прослеживает вокруг частей сценария, которые нуждаются в этом. TRACE/NET позволяет наблюдать, что делают сетевые протоколы.

trim *series*

Удаляет пробел из строки. Значение по умолчанию удаляет из головной и хвостовой частей.

series *(accepts: series port)*

/head Removes only from the head.

/tail Removes only from the tail.

/auto Auto indents lines relative to first line.

/lines Also replaces line breaks with a space.

/all Removes all whitespace.

/with

str Same as /all, but removes characters in 'str'. *(accepts: char string)*

```
str: {  
Now is the winter of our discontent made  
glorious summer by this sun of York.  
}  
print trim/auto str  
Now is the winter of our discontent made  
glorious summer by this sun of York.
```

true

Возвращает LOGIC значение для TRUE. Синонимы: YES, ON.

```
if true [print "of course"]  
of course
```

try

block

Попытка создать блок и возвращение его значения или ошибки.

block

(accepts: block)

```
if error? try [1 + "x"] [print "Did not work."]  
Did not work.  
if error? try [load "$10,20,30"] [print "No good"]  
No good
```

tuple!

```
print make tuple! [10 30 40 50]  
10.30.40.50
```

tuple?

value

Возвращает TRUE для значения tuple.

value

(accepts: any-type)

Возвращает FALSE для всех других значений.

```
print tuple? 1.2.3.4  
true  
version: 0.1.0  
if tuple? version [print version]  
0.1.0
```

type?

value

Возвращает datatype значения.

value

(accepts: any-type)

/word Returns the datatype as a word.

```
print type? 10  
integer  
print type? :type?  
native
```

unset

word

Сбрасывает значение слова.

word

Word or block of words *(accepts: word block)*

```
test: "a value"
unset 'test
print value? 'test
false
```

unset? *value*

Возвращает TRUE для значения unset.

value (*accepts: any-type*)
unset? get/any 'undefined

until *block*

Вычисляет блок, пока это не TRUE.

block (*accepts: block*)
str: "characters"
until [
 print str
 tail? str: next str
]
characters
haracters
aracters
racters
acters
cters
ters
ers
rs
s

update *port*

Обновляет данные, связанные с портом.

port (*accepts: port*)
Обновляет ввод или вывод порта. Если ожидается ввод, порт проверен для большего количества ввода. Если вывод находится на рассмотрении тогда, запись вывода.

uppercase *string*

Преобразование строки символов к верхнему регистру.

string (*accepts: any-string*)
/part Limits to a given length or position.
range (*accepts: integer any-string*)

```
print uppercase "abcdef"
ABCDEF
print uppercase/part "abcdef" 1
Abcdef
```

url!

```
print make url! "http://www.REBOL.com"
http://www.REBOL.com
```

url? *value*

Возвращает TRUE для значения url.

value (*accepts: any-type*)
Возвращает FALSE для всех других значений.
print url? http://www.REBOL.com
true
print url? "test"
false

usage

Вывод параметров командной строки.

Включает в себя варианты командной строки и примеры.

use

words body

Определяет слова, которые являются локальными в блоке.

words

Local word(s) to the block (*accepts: block word*)

body

Block to evaluate (*accepts: block*)

```
total: 1234
nums: [123 321 456]
use [total] [
  total: 0
  foreach n nums [total: total + n]
  print total
]
900
print total
1234
```

value?

value

Возвращает TRUE, если слово было установлено.

value

(*accepts: any value*)

```
test: 1234
print value? 'test
true
print value? first [test this]
true
```

wait

value

Ждет продолжительности, порта, или обоих.

value

(*accepts: number time port block*)

```
print now/time
wait 1
wait 0:00:01
print now/time
8:47:39
8:47:41
```

what

Печатает список глобально-определенных функций.

what-dir

Печатает активный путь каталога.

```
print what-dir
/C/rebol-2.0.0/
```

while

cond-block body-block

В то время как блок условия TRUE, вычисляет другой блок.

cond-block

(*accepts: block*)

body-block

(*accepts: block*)

BREAK может использоваться, чтобы выйти из блока в любой точке.

```
str: "characters"
while [ not tail? str: next str ] [
  print [ "length of" str "is" length? str ]
]
length of haracters is 9
length of aracters is 8
length of racters is 7
length of acters is 6
length of cters is 5
```

```
length of ters is 4
length of ers is 3
length of rs is 2
length of s is 1
```

word!

```
print make word! "test"
test
```

word?

value

Возвращает TRUE для значения word.

value

(accepts: any-type)

```
print word? second [1 two "3"]
true
```

write

destination value

Пишет файлу, url, или спец-порту (блок или объект).

destination

(accepts: file url object block)

value

(accepts: any value)

/binary

Preserves contents exactly.

/string

Translates all line terminators.

/direct

Opens the port without buffering.

/append

Writes to the end of an existing file.

/wait

Waits for data.

/lines

Handles data as lines.

/part

Reads a specified amount of data.

size

(accepts: number)

/with

Specifies alternate line termination.

end-of-line

(accepts: char string)

/allow

Specifies the protection attributes when created.

access

(accepts: block)

/mode

Block of above refinements.

args

(accepts: block)

/custom

Allows special refinements.

params

(accepts: block)

WRITE обычно используется, чтобы записать файл на диск, но есть много других операций, типа записи данных к URL, и т. д. (Эта функция имеет много усовершенствований, обращаться к REBOL Руководству пользователя для подробной информации.) Обычно строка или двойное значение обеспечены в этой функции, но другие типы данных, например числа или времени тоже могут быть приняты. Они будут преобразованы в текст.

```
write %junkme.txt "This is a junk file."
write %datetime.txt now
```

xor

value1 value2

Возвращает первое, особенное значение ORed со вторым.

value1

(accepts: logic char number tuple)

value2

(accepts: logic char number tuple)

Функция инфиксного оператора. Для целых чисел, каждый бит - исключительно-or'd. Для логических значений, это то же самое как не - равная функция.

```
print 122 xor 1
123
print true xor false
```

```
true
print false xor false
false
print 1.2.3.4 xor 1.0.0.0
0.2.3.4
```

yes

Аналог LOGIC TRUE.

zero?

number

Возвращает TRUE если число является нулевым.

number (*accepts: char number money time tuple*)

```
print zero? 50
false
print zero? 0
true
```