Map of RPN Calculator

RPN Calculator for Win95

1
2
3
4

Drop
Clear
Pick

<---

Depth ● Rad ○ Deg ○ Grad LastArg

Sto Rcl ▼ Purge

| Parts (4) | Probability (5) | Other (6) |
| Power (1) | Sinusoidal (2) | Hyperbolic (3) |

| X^2 | Sqrt | 10^X | Log |
| 1/X | Y^X | Exp | Ln |

+/- EEX Swap Up Down

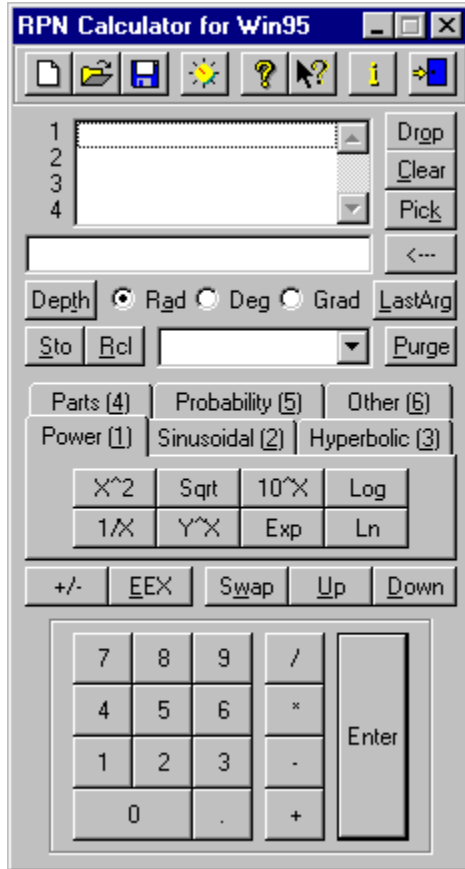| 7 | 8 | 9 | / | |
4	5	6	x	Enter
1	2	3	-	
0	.	+	

▦ What is RPN Calculator?

RPN Calculator is a tool for Windows 95 that allows to use advanced mathematical functions typical of the most powerful pocket scientific calculator.

The model to which we were inspired has been the family of the Hewlett-Packard's HP48 scientific pocket calculators, characterized by the adoption of the Reverse Polish Notation, instead of the Algebraic: this notation offers remarkable advantages in terms of management of complex expressions, since the ease with which the partial operations could be performed certainly compensates the necessary apprenticeship to use it.

For those people that don't know the RPN logic, in the present guide were included some notes. Click here, to see them.

Follows a partial list of the characteristics of the RPN Calculator:

- Accept input from keyboard safely
- 17 buttons typical of elementary RPN calculators
- 43 advanced mathematical functions
- Ability to recover the argument(s) of the last function (LastArg)
- Up to 100 memories in which you can store numbers (Variables)
- Open and Save commands, for storing the Working Environment
- AutoSave function, for keeping your data automatically
- Options for personalizing the view of the Stack

Note that HP and HP48 are trademarks of Hewlett-Packard Corporation.

🖩 Comparison between RPNCalc and HP48

This page is to be intended as a guide for HP's users to discover the differences with RPN Calculator

Follows a partial list of the differences between these products.

<u>Stack</u>
- RPNCalc's Stack is oriented with the first line up.
- HP48 provides more commands for handling stack: for example, allows to duplicate more than one line at a time, or to rotate the first three or more lines in whatever direction.
- The commands <u>Up</u> and <u>Down</u> coincide with HP's ROLLD and ROLL respectively, and not vice-versa (because of the different orientation of the Stack).
- HP48 offers two additional <u>Display Modes</u> (Sci: scientific - Eng: scientific with exponent multiple of 3)

<u>Display</u>
- HP48 allows to insert more than one number at a time.
- The contents of RPNCalc's Display can be copied to Clipboard, and you can copy numbers to Display through the Clipboard.

<u>Memory Management</u>
- RPNCalc offers an easier way to save and restore the <u>Working Environment</u> (substantially Stack, Display and Variables) and allows also to store as many environments as you want.
- RPNCalc is safer than HP48, because you have to <u>confirm</u> every deletion not fully recoverable.
- HP48 offers some functions for performing simple mathematical operations with the Variables (for example: adding a number to the content of a Variable and automatically storing the result in the same Variable).

<u>Advanced Functions</u>
- RPNCalc doesn't handle the HP48's objects that are not Real numbers:
     Complex numbers, Vectors and Matrices (Array), Units of Measurement, numbers in a whatever base (Binary, Octal, Hexadecimal).
- HP48 calculators provide some additional functions, like:
     EquationWriter (an editor for writing functions in the natural form), MatrixWriter (for entering Arrays or Matrices in a tabular form), HP Solve (find the zeros of a function), an environment for tracing graphs and for analyzing them, Statistics, Algebra and Calculus.
- HP's calculators are programmable.

Great part of the points previously listed reflect the choices characterizing this software: RPN Calculator is in fact a scientific calculator, so doesn't necessarily have to supply functions like graphics or calculus.

If you require any of such functions on PC, probably you need products like Mathematica, Maple V, MathCad and StatGraphics (for advanced statistics), or generic software like Microsoft Excel and Lotus 1-2-3.
However, some of such functions will be provided with next versions.

Note that the software mentioned in the previous lines is not necessarily good for any purpose, and that the fact of having cited it, doesn't mean that you could not find better products (depends on the usage that one wants to do with).

Note also that RPN Calculator is not in competition with Hewlett-Packard's calculators, because of the different target of these products: one is a software, dedicated to powerful and massive computers, and the other is a pocket calculator.

It's undeniable, however, that products like RPN Calculator offer a simplicity of usage unbeatable for

small calculations.

📟 The Working Environment

With *Working Environment* we intend the union of data and options that characterize the status of RPN Calculator. In other words, the W.E. is composed by the <u>Stack</u>, the <u>Display</u>, the <u>List of the Variables</u> and other information.

An important question relies on the different types of information about the status that RPN Calculator handles: the first type is constituted by the options (shown clicking on the <u>Options</u> button).

Options are characterized by the fact that they are stored in the Registry of Windows 95: so their values are common to all the sessions of RPN Calculator.

Other information that are stored in the Registry are the position of the main window, and the type of <u>angle</u>.

The second class is composed by information like the <u>Maximum Random Number</u>, the current <u>Seed</u> and the data necessary to <u>LastArg</u>: such information are strictly dependent on the session, so are saved in <u>data files</u> simultaneously with the Stack, the Display and the List of the Variables.

▦ Overview of Reverse Polish Notation

Think about a typical mathematical operation in algebraic notation, like the following
$$3 * (5 - 2) + \log(10)$$

If you want to perform it in a classical algebraic calculator (that is able to manage parenthesis), you have only to press the corresponding buttons; this is apparently the simplest way to do an operation, but it has some disadvantages, especially with big and complex expressions: these would be handled easier if could be decomposed in many simpler expressions.

Consider the mathematical expression reported above: you can decompose it in the following sub-expressions: 3 * (5 - 2), log(10).

Note that we have analyzed a deeper level of the structure: we could say that the Level 0 is the whole expression, the Level 1 is constituted by the sub-expression reported above, and so on.

Follows a figure that explains it.
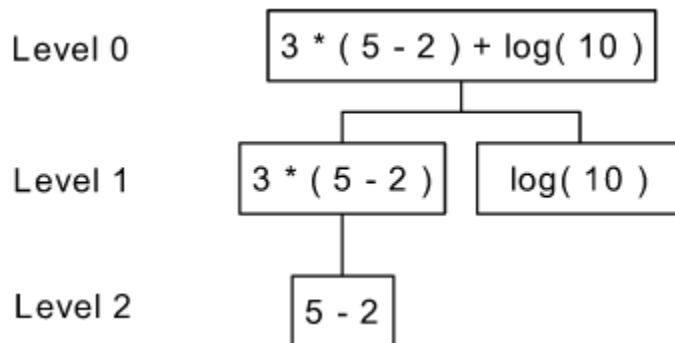
At this point we concern only with simple operations, that manipulate numbers directly; to resolve the expression you have to apply the following algorithm:
- Execute the expressions reported at the last level and save the results
- Execute the expression at the following level, using the results of the previous level
- Repeat the last point for every level

Note that the results of the Level n are used by the Level n-1.

For the example with which we started, it means that you have to do this passages:
> **Level 2**
> 5 - 2 = 3 *[store the result in memory, in a location x]*
> > **Level 1**
> > 3 * x = 3 * 3 = 9 *[store in y]*
> > log(10) = 1 *[store in z]*
> > > **Level 0**
> > > y + z = 10

The final result is 10.

The memories in which RPN Calculators stores the partial results (those that I called x, y and z) are lines of the Stack.

There is still a point uncovered: what is Reverse Polish Notation (RPN)?
It is a mathematical notation based on the concepts reported above: this notation was introduced by Lukasiewicz in 1929 and then modified by E. W. Dijkstra in 1960.

In conclusion, the equivalent expression in RPN of the algebraic reported as example is:

5 2 - 3 * 10 log +

This notation express this: "get 5 and 2; subtract these two numbers *[the result is indicated as x in the following lines]*; get 3; multiply x for 3 *[this is y]*; get 10; perform the logarithm of 10 *[this is z]*; add y and z".

Note that it expresses exactly what the previously reported algorithm does.

For better understanding these arguments, it's recommended to read about the Stack.

▦ The Stack

In the following lines, we assume that you have already read the _Overview of Reverse Polish Notation_.

Considering the same example reported in the _Overview_,
$$3 * (5 - 2) + \log(10)$$
we can remember that the corresponding expression in RPN is
$$5\ 2 - 3 * 10\ \log +$$

If you have to perform this calculation on a traditional algebraic calculator, like the one that Microsoft ships with Windows 95, you have to press the buttons in the following order:

3	*	(	5	-	2	)
	+	1	0	log	=	

The same calculation in RPN presents immediately two problems:
> - after that you have inserted the first number, 5, how can you put in the second one?
> - suppose that the previous problem is resolved, it's already undefined how can you store the partial results.

The key is the Stack.

The Stack is a LIFO (_Last In First Out_) structure: substantially it is a collection of elements (in RPNCalc, this elements are numbers), ordered according to the sequence of insertion in a such way that the last element inserted is the first that is extracted.
You can think the Stack as a list of lines, each of which contains a number.

If you want to insert a number in the Stack, you have to put it in the Display and then to copy it in the Stack pressing the Enter button: in other words, this button is the answer to the first question.
So, for inserting the number 2 after 5, you have to press the following sequence:

5	Enter	2
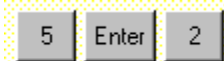
At the end of this sequence of buttons, your RPN Calculator looks like this:

1	5		▲	Drop
2				Clear
3				Pick
4			▼	
		2	<---	

Every function of RPN Calculator, before performing the operation for which is called, checks if the Display is empty: if there is a number, as in the previous figure, it is used as argument.
If you press the subtraction button, as required by the example, the result will be
$$5 - 2 = 3$$

Is time to answer to the second question: where the results are stored?
If you have already accomplished the subtraction, you know the solution: results are stored in the first line of the Stack, so they can be used as argument for the next functions.
Note that this is exactly what we have said in the _Overview_ about Levels: the results of Level n are the arguments of Level n-1.

Now you can complete the calculation by yourself. For seeing the right combination of buttons, click here.

RPN Calculator implements some commands to handle the Stack; if you need to know something about one of them, click on the corresponding link:

Drop, Clear, Pick, Depth, Swap, Up, Down.

🔢 The Display

The display is the line of the RPN Calculator in which you can edit the numbers before inserting it in the Stack

The Display is a common edit box, so you can perform any typical operation of these controls: for example, you can select a part of its content and then copy it in the Clipboard (for doing that, after selecting what you want to copy, click on the right button of the mouse upon it and choose "Copy"), or you can insert some digits simply positioning the flashing bar (caret) where you want to put in and then starting to press on your keyboard.

Note that there are two ways to insert a number: the first require the mouse, and consist simply in clicking on the buttons that you choose; the second way privilege the use of the keyboard.
The first one is more comfortable but less fast than the second one.

There are some question to know if you want to use the keyboard:
- you have to use the numeric keypad for performing one of the four basic functions (add, subtract, multiply and divide)
- there is a shortcut for every command and function: to know about one of them consult the context sensitive help by clicking on the What's This (Shift+F1) button

The Display is (or should be) error proof, because it doesn't allow to insert some erroneous combinations:
- you can't insert alphabetical chars or symbols, except 'E' and '-'
- you can insert only one 'E' char
- you can insert only one '-' char ahead of the mantissa and one before the exponent

- if you insert, for instance, the sequence [1] [EEX] and then you click on the [+/-] button, RPN Calculator will not change the sign of the display as requested, because the "+/-" button works on the mantissa only if you have not yet put in the 'E' char; if you have already inserted the 'E' char, the "+/-" button works only on the exponent, that in the considered example does not exist. If you want to insert the minus sign before the mantissa, after that you have already inserted the 'E' char, you have to position the flashing bar (caret) at the beginning of the number, and then press the '-' button on your keyboard.

Functions and Commands

It's hard to say what is the difference between functions and commands, because both process data and return a result.

For example, you can consider the function 'Sin': it takes a number from the <u>Stack</u> and return the sin of this number; on the other hand, the command 'Swap' takes two numbers from the Stack, and then interchange them.

These examples underline the most important distinction between functions and commands: one works on the lines of the Stack supposing that those lines consist of numbers, and the other don't value the content of the lines.
Functions require numeric data and return numbers, while commands could work on chars or binary sequences without problems, no matter the contents of the lines of the Stack.

There is also a logistic separation between these classes: in fact functions are collected in the two sites reported below.
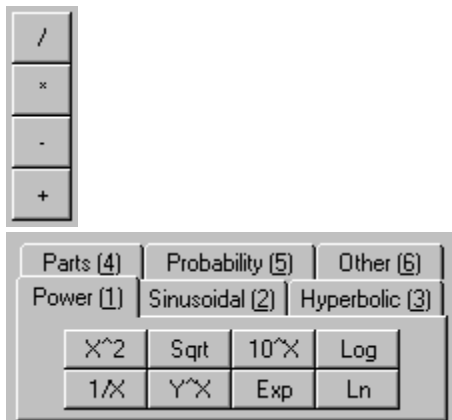
The first group is composed by the fundamental mathematical operations (addition, subtraction, multiply and division), so are located near the numeric digits (as in every other calculator), while the second group is constituted by advanced functions.

🗐 Data Files

One of the earliest implemented feature of the advanced scientific calculator was a permanent memory, in which you can store variables, constants and, why not, simple routines.

Although RPN Calculator can't yet handle a programming language, there are many situations in which the ability to save your data for later recovery is very important: for example, if you want to save a set of constants or parameters that you need frequently, or if you want to resume later a calculation that you have to break.

In RPN Calculator there are two ways to store your data:
- directly, through the Save command
- indirectly, through the AutoSave feature

The Save Command

First of all, we consider the Save command: it is accessible by clicking on this button of the toolbar 🖫 , after that will be shown the common dialog box "Save as" in which you are asked to insert the name of the file
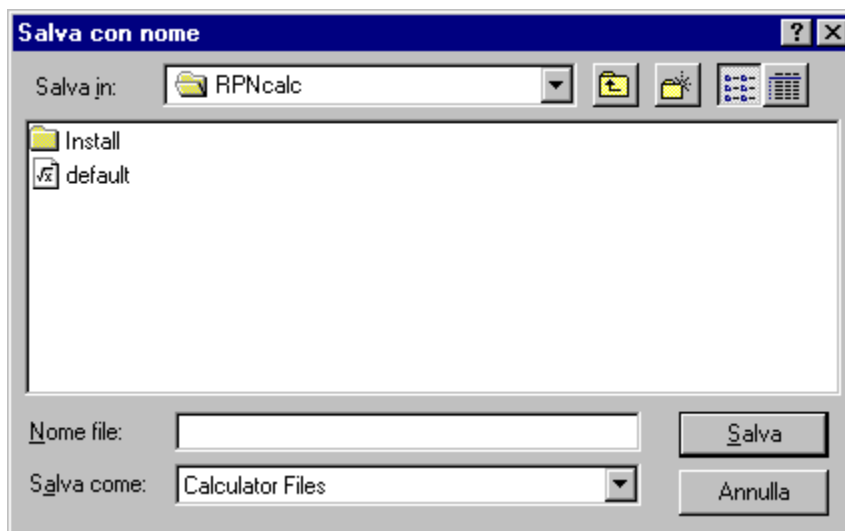
Sorry, I have the italian version of Windows 95, but whatever version of this operative system you have, the "Save as" dialog box will not be too different.

After that you have chosen the file that you want to save, if this file already exists you will asked to confirm if you want to overwrite it.
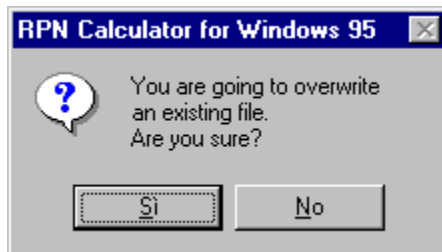
If you definitely choose to, the file will be overwritten and your data will be saved.

For later recovering the data stored in every file you want, click on the Open button .

The AutoSave feature

Surely it's important to have the ability to store your data when you need to, but sometimes is more useful if the application does it automatically: RPN Calculator has the AutoSave option for this.

If you enable AutoSave, RPN Calculator will behave in a such way:
          - on closing, it stores automatically your data in a file choosen in advance
          - on restarting, it will reload your data, so that you will have your environment exactly as you left it

Click here, for more information on how to enable and configure AutoSave.

What is stored in a Data File?

Follows the complete list of the items that are stored in such files:
          - the whole List of the Variables
          - the Stack
          - the Display
          - the maximum random number
          - the last seed for random number generation
          - the arguments of the last function (for LastArg)

▥ Options

is the button of the toolbar that allows to open the dialog through which you can set the options.

There are three classes of options:

Help System

    Tooltips
        Allows to disable or enable tooltips.

Display Modes

    Auto
        If you select this option, numbers are shown in a scientific form (ex.: 1E5) or not, depending on what *Precision* you have choose.
    Fix
        Numbers are shown with a fixed number of decimals, determined by *Precision*.
    Precision
        If *Auto* is selected, set how many digits the <u>mantissa</u> has; otherwise establish how many digits has the fractional part of the number.

AutoSave

    AutoSave is a feature that perform the automatic storing, on exit, of the <u>Stack</u> and of the <u>Variables</u> in the *Default File*; with *AutoSave* enabled, RPN Calculator also loads automatically the *Default File*.
    <u>Click here</u> for further information on data files.

    Enable
        If checked, the *AutoSave* is active.

    Default File
        Here you must insert the file that RPN Calculator has to use for *AutoSave*.

    Browse...
        Allows to select the Default File by browsing your system with the common open dialog box.

The button *Register* is not yet enabled, because it concern only the final releases.

▥ Management of Variables

Even the cheapest calculator is able to remember a number during a session for later elaboration. The most expensive ones can store more than one number at the same time, and can remember those numbers even if you turn off the calculator.

RPN Calculator can store up to 100 numbers in as many *Variables*, that you can save later in <u>data files</u>. Every Variable is named with a label of up to 20 chars of whatever type: you can use letters, digits or other symbols.

You can perform a limited set of operations on Variables: you can create a new Variable or update an existing one, copy the content of a Variable in the Stack, delete a Variable or the whole List of the Variables, and naturally you can save the List of the Variables with a data file.

The objects specifically designed for handling the Variables are grouped in the following line:

Follows a description of step-by-step procedure for accomplishing these operations.

Create a new Variable
- Select the line of the Stack that you want to store in a Variable
- Press the <u>Sto</u> button
- Insert the name of the new Variable in the text box
- Press the OK button

ATTENTION! If the Variable already exists, this procedure will update it and therefore its older content will be lost.

Update an existing Variable
- Select the line of the Stack that you want to store in a Variable
- Press the Sto button
- Insert the name of the Variable in the text box (note that you can do that simply, by clicking on the drop-down arrow near the text box and then choosing one of the names in the list)
- Press the OK button

ATTENTION! The older content of the Variable will be lost.

Copy the content of a Variable in the Stack
- Click on the drop-down arrow between the <u>Rcl</u> and the <u>Purge</u> buttons
- Select from the list the name of the Variable whose content you need to store in the Stack
- Press the Rcl button

Delete one Variable or the whole List of the Variables
- Click on the drop-down arrow between the Rcl and the Purge buttons
- Select from the list the name of the Variable that you want to delete (note that you must select a Variable even if you have to delete all the Variables)
- Press the Purge button
- Select the option that you prefer (deleting one or all the Variables)
- Press the OK button

ATTENTION! If you choose to delete all the Variables, you will NOT be able to recover them, neither with <u>LastArg</u>. If you delete only one Variable, you will be able to recover only the content.

<u>Save the List of the Variables with a data file</u>
       - Click on the <u>Save</u> button

Note that these instructions are written particularly for the mouse, but you can accomplish the same procedure using the keyboard.

RPN Calculator restrict himself to assume some aspects of the user interface and to implement part of the functions of HP's calculators through data structures and algorithms produced by original work.
The differences between RPN Calculator and HP's are substantial rather than of appearance: so, there are some important distinctions of behavior between the two systems. If you want a list of such differences, click here.

There is an exception to this rule: if you decide to <u>Purge</u> Variables, clicking on the appropriate button, RPN Calculator will show a dialog box in which you must choose if you want to delete the current Variable, the whole <u>List of the Variables</u>, or to renounce.
If you conclude to delete one or all the Variables, it will be done immediately.

Note also that if you choose to <u>Drop</u> one line of the Stack, RPNCalc don't ask you for confirm, because you can reverse this operation with <u>LastArg</u>.

| 5 | Enter | 2 | - | 3 | × | 1 | 0 | Log | + |

Though the expression of the example is very simple, you can see that the Reverse Polish Notation is more concise than Algebraic: in fact it requires a sequence of 10 mouse clicks instead of 12.

This list contains all the Variables, so it's called "the List of Variables".

Parts (4)	Probability (5)	Other (6)
Power (1)	Sinusoidal (2)	Hyperbolic (3)

X^2	Sqrt	10^X	Log
1/X	Y^X	Exp	Ln

Parts (4)	Probability (5)	Other (6)	
Power (1)	Sinusoidal (2)	Hyperbolic (3)	

Sin	Cos	Tan	
ASin	ACos	ATan	

Parts (4)	Probability (5)	Other (6)
Power (1)	Sinusoidal (2)	Hyperbolic (3)

SinH	CosH	TanH	
ASinH	ACosH	ATanH	

Power (1)	Sinusoidal (2)	Hyperbolic (3)	
Parts (4)	Probability (5)		Other (6)
Abs	Sign	Int	Frac
Round	Floor	Ceil	Mod

Power (1)	Sinusoidal (2)	Hyperbolic (3)
Parts (4)	Probability (5)	Other (6)

Comb	Perm	!	
Rand	Seed	->MaxR	MaxR->

Power (1)	Sinusoidal (2)	Hyperbolic (3)
Parts (4)	Probability (5)	Other (6)

Pi	%	%CH	%T
->HMS	HMS->	HMS+	HMS-

The edit box in which you have just clicked is the Display.

Shortcuts are sequences of keystrokes that allows to execute a command or a function faster than switching between the buttons with the tabs; when you will became experienced, you will be able to perform complex operation with the keyboard quicker than by using the mouse, because of the shortcuts.

The mantissa is the part of the number in scientific form that is at the left of the 'E' letter.
The exponent is the part of the number in scientific form that is at the right of the 'E' letter.
For example, in the following Display the mantissa is 1.234 and the exponent is 56.

```
1.234E56
```

New

Clear the <u>Working Environment</u>

Keyboard Shortcut: CTRL+N

Input:
　　None

Output:
　　None

Remarks:
　　Clear the <u>Stack</u>, delete the contents of the <u>Display</u> and purge any <u>Variable</u>.

Open

Open a file

Keyboard Shortcut: CTRL+O

Input:
> None

Output:
> None

Remarks:
> Shows a dialog box in which you must select the file to load; then, if you choose to, loads it.
> Note that will be affected the Stack, the Display and the List of the Variables.
> For further information on Data Files, click here.

🖬 Save

Save the <u>Working Environment</u> in a file

Keyboard Shortcut: CTRL+S

Input:
None

Output:
None

Remarks:
Shows a dialog box in which you must select the name of the file to save (update); then, if you choose to, save the Working Environment in such file.
For further information on Data Files, <u>click here</u>.

Exit

Exit

Keyboard Shortcut: ALT+F4

Input:
None

Output:
None

Remarks:
If AutoSave is enabled, save the Working Environment in the default file, and then exit.

+/-

Changes the sign of a number

Keyboard Shortcut: CTRL+I
> Note that "I" stands for sign inversion.

Input:
> The content of the <u>Display</u>, if it is not empty, otherwise the first line of the <u>Stack</u>.

Output:
> If the Display is not empty, change the sign of it, otherwise change the sign of the first line of the Stack.

Remarks:
> If the Display and also the Stack are empty, do nothing.
> If in the Stack there is the <u>E char</u>, change the sign of the exponent.

<---

Delete the contents of the <u>Display</u>

Keyboard Shortcut: ALT+X

Input:
None

Output:
None

Remarks:
Note that the contents of the Display will be lost in a irrecoverable way.

![icon] Clear

Clear the <u>Stack</u>

Keyboard Shortcut: ALT+C

Input:
　　None

Output:
　　None

Remarks:
　　Note that the contents of the Stack will be lost in a irrecoverable way.

▦ **Depth**

Number of lines of the Stack

Keyboard Shortcut: ALT+T

Input:
> None

Output:
> Number of lines of the Stack (excluding this output).

Remarks:
> None

🔳 **Down**

Roll down the <u>Stack</u>

Keyboard Shortcut: ALT+D

Input:
    None

Output:
    None

Remarks:
    Rotate the Stack toward down (out the bottom and in the top).

🖩 Drop

Delete the first line of the <u>Stack</u>

Keyboard Shortcut: ALT+O

Input:
The first line of the Stack.

Output:
None

Remarks:
None

▤ **EEX**

Insert the 'E' char in the <u>Display</u>

Keyboard Shortcut: ALT+E

Input:
　　None

Output:
　　None

Remarks:
　　The 'E' char stands for '10^'; it is used for the inserting numbers in scientific form.

Enter

Enter or duplicate

Keyboard Shortcut: SHIFT-ENTER
Use it if another button has the focus (i.e. has a border around the text).

Input:
The content of the Display, if it is not empty, otherwise the first line of the Stack.

Output:
If the Display is not empty, insert it at the beginning of the Stack, otherwise duplicate the first line of the Stack.

Remarks:
None

▤ LastArg

Last argument(s)

Keyboard Shortcut: ALT+L

Input:
    None

Output:
    The argument(s) of the last function.

Remarks:
    None

▥ **Pick**

Pick a line of the <u>Stack</u>

Keyboard Shortcut: ALT+K

Input:
　　The selected line of the Stack.

Output:
　　Insert the input line at the top of the Stack.

Remarks:
　　The selected line is kept in the Stack.
　　If the Stack is empty, do nothing.

▥ Purge

Delete <u>Variables</u>

Keyboard Shortcut: ALT+P

Input:
   None

Output:
   None

Remarks:
   Delete one or all the variables: if you select this command, RPN Calculator shows a window in which you can choose to delete the selected Variable, to clear all Variables or to cancel this action.
   If you choose to delete one variable, the content will be saved for recovering with <u>LastArg</u>.
   This command is unavailable until you select a Variable in the List of the Variables.
   Note that the deleted Variable(s) will be irrecoverable.

▥ **Rad, Deg & Grad**

Angle mode

Keyboard Shortcut: ALT+A

By using it, you select the Radians mode; you can switch to the other modes using the cursor keys.

Input:

None

Output:

None

Remarks:

These affect sinusoidal functions (<u>Sin</u>, <u>Cos</u>, <u>Tan</u>, <u>ASin</u>, <u>ACos</u> and <u>ATan</u>).

⊞ Rcl

Recall Variable

Keyboard Shortcut: ALT+R

Input:
None

Output:
Content of the Variable selected in the List of the Variables.

Remarks:
This command is unavailable until you select a Variable in the List of the Variables.

▤ Sto

Store <u>Variable</u>

Keyboard Shortcut: ALT+S

Input:
The selected line of the <u>Stack</u>.

Output:
None

Remarks:
If you select this command, RPN Calculator shows a window in which you must insert the name of the Variable: if this is already used, the corresponding Variable is updated, and the old value will be lost. The name of the Variable can be long max. 20 characters; such characters can be of any type (upper or lower case, digits, symbols, etc.).
This command is unavailable until you select a line of the Stack.
Note also that you can define a maximum of 100 Variables.

▥ Up

Roll up the <u>Stack</u>

Keyboard Shortcut: ALT+U

Input:
   None

Output:
   None

Remarks:
   Rotate the Stack toward up (out the top and in the bottom).

Swap

Interchange the first 2 lines of the Stack

Keyboard Shortcut: ALT+W

Input:
The first two lines of the Stack.

Output:
The content of the 2nd line in the 1st and vice-versa.

Remarks:
None

+

Add

Keyboard Shortcut: "+" key of the numeric keypad

Input:
   The first two lines of the Stack.

Output:
   The sum of the two numbers.

Remarks:
   The arguments are saved for recovering with <u>LastArg</u>.

▯ -

Subtract

Keyboard Shortcut: "-" key of the numeric keypad

Input:
The first two lines of the Stack.

Output:
[2nd line] - [1st line]

Remarks:
The arguments are saved for recovering with <u>LastArg</u>.

*

Multiply

Keyboard Shortcut: "*" key of the numeric keypad

Input:
The first two lines of the Stack.

Output:
[1st line] * [2nd line]

Remarks:
The arguments are saved for recovering with <u>LastArg</u>.

/

Divide

Keyboard Shortcut: "/" key of the numeric keypad

Input:
The first two lines of the Stack.

Output:
[2nd line] / [1st line]

Remarks:
The arguments are saved for recovering with LastArg.

ACosH

Arc hyperbolic cosine

Input:
> The first line of the Stack.

Output:
> arccosh([1st line])

Remarks:
> The argument is saved for recovering with <u>LastArg</u>.

ASinH

Arc hyperbolic sine

Input:
The first line of the Stack.

Output:
arcsinh([1st line])

Remarks:
The argument is saved for recovering with LastArg.

ATanH

Arc hyperbolic tangent

Input:
      The first line of the Stack.

Output:
      arctanh([1st line])

Remarks:
      The argument is saved for recovering with <u>LastArg</u>.

▥ CosH

Hyperbolic cosine

Input:
The first line of the Stack.

Output:
cosh([1st line])

Remarks:
The argument is saved for recovering with LastArg.

▦ SinH

Hyperbolic sine

Input:
The first line of the Stack.

Output:
sinh([1st line])

Remarks:
The argument is saved for recovering with <u>LastArg</u>.

▥ TanH

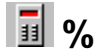*Hyperbolic tangent*

Input:
The first line of the Stack.

Output:
tanh([1st line])

Remarks:
The argument is saved for recovering with LastArg.

%

Percent

Input:
The first two lines of the Stack.

Output:
[2nd line] * [1st line] / 100

Remarks:
The arguments are saved for recovering with LastArg.

▤ %CH

Percent change

Input:
> The first two lines of the Stack.

Output:
> 100 ([1st line] - [2nd line]) / [2nd line]

Remarks:
> The arguments are saved for recovering with <u>LastArg</u>.

%T

Percent of total

Input:
The first two lines of the Stack.

Output:
100 [1st line] / [2nd line]

Remarks:
The arguments are saved for recovering with LastArg.

▦ HMS-> [->HMS]

Hour Minute Seconds to decimal [vice versa]

Input:
    The first line in HMS form [in decimal form]

Output:
    The first in decimal form [in HMS form]

Remarks:
    A number in HMS format is a number H.MMSSs, where:
    - H indicates hour, and is composed by zero or more digits;
    - MM are two digits in the range 0-60, and represents minutes;
    - SS are two digits in the range 0-60, and represents seconds;
    - s is zero or more digits, and represents the decimal fractional part of seconds.
    Note that the HMS format is useful not only with hours, but also with angles.
    The argument is saved for recovering with <u>LastArg</u>.

HMS+ and HMS-

Addition and Subtraction of numbers in HMS form

Input:
　　The first two lines in HMS form

Output:
　　The addition or subtraction in HMS form

Remarks:
　　A number in HMS format is a number H.MMSSs, where:
　　- H indicates hour, and is composed by zero or more digits;
　　- MM are two digits in the range 0-60, and represents minutes;
　　- SS are two digits in the range 0-60, and represents seconds;
　　- s is zero or more digits, and represents the decimal fractional part of seconds.
　　Note that the HMS format is useful not only with hours, but also with angles.
　　The argument is saved for recovering with LastArg.

▥ Pi

Greek Pi

Input:
None

Output:
Greek Pi

Remarks:
This constant is ever expressed in radians.

🔢 Abs

Absolute value

Input:
> The first line of the Stack.

Output:
> abs([1st line])

Remarks:
> The argument is saved for recovering with <u>LastArg</u>.

▦ Ceil

Smallest integer greater than argument

Input:
The first line of the Stack.

Output:
ceil([1st line])

Remarks:
The argument is saved for recovering with <u>LastArg</u>.

▥ **Floor**

Greatest integer smaller than argument

Input:
> The first line of the Stack.

Output:
> floor([1st line])

Remarks:
> The argument is saved for recovering with <u>LastArg</u>.

📟 Frac

Fractional part

Input:
>The first line of the Stack.

Output:
>frac([1st line])

Remarks:
>The argument is saved for recovering with <u>LastArg</u>.

⊞ Int

Integer part

Input:
The first line of the Stack.

Output:
int([1st line])

Remarks:
Return the argument truncated, without the fractional part.
The argument is saved for recovering with <u>LastArg</u>.

▥ **Mod**

Modulo

Input:
>The first two lines of the Stack.

Output:
>Remainder of [2nd line] / [1st line].

Remarks:
>The arguments are saved for recovering with LastArg.

▥ **Round**

Round

Input:

The first line of the Stack.

Output:

The argument rounded to closest integer

Remarks:

The argument is saved for recovering with <u>LastArg</u>.

▥ Sign

Sign

Input:
The first line of the Stack.

Output:
sign([1st line])

Remarks:
Return +1 if the argument is positive, 0 if is equal to zero, and -1 if is negative.
The argument is saved for recovering with LastArg.

1/x

Inverse

Input:
The first line of the Stack.

Output:
1 / [1st line]

Remarks:
If X is zero, the output is INF.
The argument is saved for recovering with LastArg.

🖩 **10^X**

Common antilogarithm

Input:
> The first line of the Stack.

Output:
> 10 ^ [1st line]

Remarks:
> The argument is saved for recovering with LastArg.

Exp

Exponential

Input:
The first line of the Stack.

Output:
e ^ [1st line], where 'e' is the Nepero's constant.

Remarks:
The argument is saved for recovering with LastArg.

Ln

Natural logarithm

Input:

The first line of the Stack.

Output:

ln([1st line])

Remarks:

The argument must be positive, otherwise the output is NAN.
The argument is saved for recovering with LastArg.

▥ Log

Common Logarithm

Input:
The first line of the Stack.

Output:
log([1st line])

Remarks:
The argument must be positive, otherwise the output is NAN.
The argument is saved for recovering with <u>LastArg</u>.

⬛ Sqrt

Square Root

Input:
The first line of the Stack.

Output:
Square root of the argument.

Remarks:
The argument is saved for recovering with LastArg.

X^2

Square

Input:
The first line of the Stack.

Output:
[1st line] ^ 2

Remarks:
The argument is saved for recovering with <u>LastArg</u>.

Y^X

Power

Input:
The first two lines of the Stack.

Output:
[2nd line] ^ [1st line]

Remarks:
Note that this function is useful also if you want a root of whatever radix: you have only to put the inverse of the radix desired as 1st line of the Stack.
The arguments are saved for recovering with LastArg.

ACos

Arc cosine

Input:
>   The first line of the Stack.

Output:
>   arccos([1st line])

Remarks:
>   The result is affected by the <u>angle mode</u>.
>   The argument is saved for recovering with <u>LastArg</u>.

ASin

Arc sine

Input:
The first line of the Stack.

Output:
arcsin([1st line])

Remarks:
The result is affected by the <u>angle mode</u>.
The argument is saved for recovering with <u>LastArg</u>.

▥ ATan

Arc tangent

Input:
The first line of the Stack.

Output:
arctan([1st line])

Remarks:
The result is affected by the angle mode.
The argument is saved for recovering with LastArg.

🔢 Cos

Cosine

Input:
The first line of the Stack.

Output:
cos([1st line])

Remarks:
The result is affected by the underline angle mode.
The argument is saved for recovering with LastArg.

🔢 **Sin**

Sine

Input:
The first line of the Stack.

Output:
sin([1st line])

Remarks:
The result is affected by the angle mode.
The argument is saved for recovering with LastArg.

▥ Tan

Tangent

Input:
> The first line of the Stack.

Output:
> tan([1st line])

Remarks:
> The result is affected by the <u>angle mode</u>.
> The argument is saved for recovering with <u>LastArg</u>.

!

Factorial

Input:
The first line of the Stack.

Output:
[1st line]!

Remarks:
If the argument is an integer, the factorial is calculated exactly; otherwise the output is an approximation of the value of Gamma([1st line] + 1): in this circumstance RPN Calculator uses the Stirling's asymptotic series arrested at the 10th order.
Note that X must be non negative.
The argument is saved for recovering with LastArg.

▦ Comb

Combinations

Input:
The first two lines of the Stack.

Output:
comb([2nd line], [1st line])

Remarks:
The content of the 2nd line must be greater or equal to content of the first.
The arguments are saved for recovering with <u>LastArg</u>.

▥ MaxR-> [->MaxR]

Maximum random number

Input:

For MaxR->, no input is need; the other function uses the first line of the Stack

Output:

MaxR-> return the maximum random number in the first line of the Stack; the other function has no output.

Remarks:

They allow to set (->MaxR) and recover (MaxR->) the maximum random number generated.
Initially the max random number is set to 1.
The argument is saved for recovering with <u>LastArg</u> (only for ->MaxR).

🔢 Perm

Permutations

Input:
The first two lines of the Stack.

Output:
perm([2nd line], [1st line])

Remarks:
The content of the 2nd line must be greater or equal to content of the first.
The arguments are saved for recovering with LastArg.

▦ Rand

Generate a random number

Input:
None

Output:
A random number in the 1st line of the Stack.

Remarks:
The output is affected by the maximum random number set by ->MaxR and by the seed.

Seed

Insert a seed for random numbers

Input:
The first line of the Stack.

Output:
None

Remarks:
Seed allows to set the starting value for the random numbers generator; the function Rand generates a new number, which will be substitute the current seed for the next random-number generation.
Initially, if the Working Environment is clear, the seed is set to 1.
The argument is saved for recovering with LastArg.

Information not available.

New

Clear the Stack and the List of the Variables.
The equivalent keystrokes are CTRL+N.
Click here for further information.

Open

Load an RPN Calculator file (.rcf) in memory.
The equivalent keystrokes are CTRL+O.
Click here for further information.

Save

Save current data in an RPN Calculator file (.rcf).
The equivalent keystrokes are CTRL+S.
Click here for further information.

Options

Set options like the appearance or the AutoSave function.
The equivalent keystrokes are CTRL+ALT+O.
Click here for further information.

Help index

Display the index of the On-line Help.
The equivalent keystroke is F1.

Context sensitive help

Show a short help for the selected object.
The equivalent keystrokes are SHIFT+F1.

About

Display information about RPN Calculator.

Exit

Save data, if AutoSave is enabled, and <u>exit</u>.
The equivalent keystrokes are ALT+F4.
<u>Click here</u> for further information.

Stack

It is a list of memories used for seeing and handling data. It is arranged on levels (1, 2, 3, 4, etc.). Although are visible only 4 of such levels, the Stack contains 1000 of them. It is the basis of the RPN logic: you must enter the arguments in the Stack before selecting the operations to do with.
Click here for further information.

Drop

Delete the selected or, if none does exist, the first line of the Stack.
The equivalent keystrokes are ALT+O.
Click here for further information.

Clear

Clear the contents of the Stack.
The equivalent keystrokes are ALT+C.
Click here for further information.

Pick

Copy the selected line of the Stack in X.
The equivalent keystrokes are ALT+K.
Click here for further information.

Display

It is a line in which you can insert data through keyboard and/or by clicking on the buttons of the RPN Calculator.
Click here for further information.

<---

Delete the contents of the Display.
The equivalent keystrokes are ALT+X.
<u>Click here</u> for further information.

Depth

Return the number of lines of the Stack.
The equivalent keystrokes are ALT+T.
Click here for further information.

Rad, Deg & Grad

Set the angle mode.
The equivalent keystrokes are ALT+A (this select Rad, so you can choose Deg or Grad simply using the cursor keys).
Click here for further information.

LastArg

Return the argument(s) of the last function.
The equivalent keystrokes are ALT+L.
Click here for further information.

Sto

Save the selected line of the Stack in a Variable. Note that is required that a line of the Stack is selected. It shows a dialog box in which you must enter the name of the variable that you want to create or update.
The equivalent keystrokes are ALT+S.
Click here for further information.

Rcl

Insert in the Stack the contents of the variable shown in the List of the Variables.
The equivalent keystrokes are ALT+R.
Click here for further information.

List of the Variables

It is a collection of memories, each of which has to be called with a name of maximum 20 characters: in whole the documentation such memories are referred as *Variables*. You can define maximum 100 Variables. Use Sto, Rcl and Purge to handle Variables.
Click here for further information.

Purge

Delete one variable or clear the whole List of the Variables.
The equivalent keystrokes are ALT+P.
<u>Click here</u> for further information.

+/-

Change the sign of the Display, if it is not empty, or the sign of the first line of the Stack.
The equivalent keystrokes are CTRL+I.
Click here for further information.

Swap

Interchange the first two lines of the Stack.
The equivalent keystrokes are ALT+W.
Click here for further information.

Up

Roll up the Stack (out the top and in the bottom).
The equivalent keystrokes are ALT+U.
<u>Click here</u> for further information.

Down

Roll down the Stack (out the bottom and in the top).
The equivalent keystrokes are ALT+D.
Click here for further information.

Enter

Insert in the Stack the contents of the Display, if any, or duplicate the first line of the Stack.
You must use SHIFT+ENTER if another button has the focus (i.e. has a border around the text).
Click here for further information.

Power Functions

Set of functions that concern powers and exponential.
Follows the list of these functions:
X^2, Sqrt, 10^X, Log, 1/X, Y^X, Exp, Ln.

The equivalent keystrokes are defined in this way:
the first function corresponds to CTRL+1, the second corresponds to CTRL+2 and so forth up to the last function to which corresponds CTRL+8.

Sinusoidal Functions

Follows the list of these functions:
Sin, Cos, Tan, ASin, ACos, ATan.

The equivalent keystrokes are defined in this way:
the first function corresponds to CTRL+1, the second corresponds to CTRL+2 and so forth up to the last function to which corresponds CTRL+8.

Hyperbolic Functions

Follows the list of these functions:
SinH, CosH, TanH, ASinH, ACosH, ATanH.

The equivalent keystrokes are defined in this way:
the first function corresponds to CTRL+1, the second corresponds to CTRL+2 and so forth up to the last function to which corresponds CTRL+8.

Parts Functions

Set of functions that concern absolute values, integer and fractional parts, remainder and others.
Follows the list of these functions:
Abs, Sign, Int, Frac, Round, Floor, Ceil, Mod.

The equivalent keystrokes are defined in this way:
the first function corresponds to CTRL+1, the second corresponds to CTRL+2 and so forth up to the last function to which corresponds CTRL+8.

Probability Functions

Set of functions that concern probability and random number generation.
Follows the list of these functions:
Comb, Perm, !, Rand, Seed, ->MaxR and MaxR->.

The equivalent keystrokes are defined in this way:
the first function corresponds to CTRL+1, the second corresponds to CTRL+2 and so forth up to the last function to which corresponds CTRL+8.

Other Functions

Set of constants and functions that don't belong to other categories .
Follows the list of these functions:
Pi, %, %CH, %T, HMS conversion functions, HMS+ and HMS-.

The equivalent keystrokes are defined in this way:
the first function corresponds to CTRL+1, the second corresponds to CTRL+2 and so forth up to the last function to which corresponds CTRL+8.