

ValCalc 1.0

COLLABORATORS

	<i>TITLE :</i> ValCalc 1.0		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		June 8, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	ValCalc 1.0	1
1.1	In the beginning...	1
1.2	Its not my fault...	1
1.3	Spread the word AND the program	2
1.4	Cardware ? What the heck is Cardware	2
1.5	So, what is it good for ? Absolutely nothing - say that again	2
1.6	'How do I install it ?' you might ask	3
1.7	Can I use it ?	3
1.8	A bug ? That must be a mistake	3
1.9	Now where did that red rectangle come from ?	3
1.10	eval ?	4
1.11	Calculator ?	4
1.12	Isn't that what eval does ?	4
1.13	Hmm, I wonder what this button does...	4
1.14	That looks nice	5
1.15	Let's see, where can I find Quit	5
1.16	"	6
1.17	"	6
1.18	"	6
1.19	"	6
1.20	Nothing ever stays the same	7
1.21	It's a long shot and Mantle runs for the third base...	7
1.22	The values they are a-changing	7
1.23	First we must have a base	8
1.24	...and when Nina saw this, she got a funny expression on her face	8
1.25	Express yourself	8
1.26	Let us try to follow the correct procedure	10
1.27	"	11
1.28	"	12
1.29	"	12
1.30	VALstory	13
1.31	...without whom this would all have been possible...	13

Chapter 1

ValCalc 1.0

1.1 In the beginning...

ValCalc 1.0 Copyright (C) 1995 by Sune Lindhe

First a little note:

In this manual 10^3 means $10*10*10$, 3^2 means $3*3$ and so on.

Disclaimer	The usual "it's not my fault"-information.
Distribution	Distribution conditions.
Cardware	Cardware information.
Requirements	What you need to use ValCalc.
Introduction	A short description of ValCalc.
Installation	How to install ValCalc.
How to use ValCalc	How to use ValCalc.
Bug reports	How to report a bug in ValCalc.
Todo	What I think should be added/improved.
History	History of releases.
Acknowledgements	...And now I would like to thank the following people...

1.2 Its not my fault...

Disclaimer

I hereby reject any liability or responsibility for these or any other consequences from the use of ValCalc whatsoever. This includes, but is not limited to, damage to your equipment, to your data, personal injuries, financial loss or any other kinds of side effects.

Although ValCalc has been tested thoroughly on several different machines, I cannot rule out the possibility that ValCalc is somehow incompatible to your equipment, has bugs that show up on your equipment or does not do what it is supposed to do on your equipment.

It is your responsibility to take any precautions necessary to

protect yourself from these or any other effects. I explicitly reject any liability or responsibility from the consequences of you using ValCalc.

Sune Lindhe

1.3 Spread the word AND the program

Distribution

ValCalc may be distributed under the following circumstances:

- o Profits may not be made by distributing this piece of software.
- o Only a nominal fee for costs of magnetic media should be charged.
- o If included on CD collections, the CD should be for the public domain.
- o The executable and documentation should be distributed together.

1.4 Cardware ? What the heck is Cardware

Cardware

Cardware means that I would like you to send me a card (or e-mail), if you are using this program, if you have any comments/ideas to ValCalc or, in the unlikely event that you find a bug, a Bug report.

If you mail me, and have an e-mail address, you will be registred and recieve updates to ValCalc whenever they become available.

My adressess are:

Sune Lindhe
Boyesgade 5 st th
1622 KBH V
Denmark

Internet:

ec928910@ebar.dtu.dk This adress will at least be valid until ultimo 1997

1.5 So, what is it good for ? Absolutely nothing - say that again

Introduction

I will try to make this short. The heart of ValCalc is a little calculator capable of evaluating standard mathmatical expressions with the standard trigonometric functions build-in. ValCalc can be executed from the CLI or Workbench.

When started from the CLI with an argument, it will evaluate the argument and print the result to the CLI.

When started from the CLI without an argument or from Workbench, ValCalc will open a window which looks pretty much like a standard desk-calculator. In this 'mode' ValCalc is capable of many things with the most powerfull being

the ability to load and execute procedures written by you.
Being able to be used in these two ways, ValCalc replaces both eval and Calculator.
For information about using ValCalc see How to use ValCalc

1.6 'How do I install it ?' you might ask

Installation

This I know I can keep short. Copy ValCalc to whatever drawer you want it to be in.

1.7 Can I use it ?

Requirements

- o Kickstart 2.0 or better
- o And of course: The only computer that makes it possible

1.8 A bug ? That must be a mistake

Bug Report

If you should discover a bug I would be very pleased if you would send me a description of bug, the circumstances in which it appeared and decription of your system eg. model, kickstart, memory, cpu and so on.
My adreses are:

Sune Lindhe
Boyesgade 5 st th
1622 KBH V
Denmark

Internet:

ec928910@ebar.dtu.dk This adress will at least be valid until ultimo 1997

1.9 Now where did that red rectangle come from ?

How to use ValCalc

ValCalc can used in two ways:

- CLI Mode Use this mode to evaluate a single expression.
 - GUI Mode Use this mode if you would like to evaluate more than one expression and be able to use all the fancy stuff in ValCalc.
-

Important note: ValCalc requires a stack of at least 100000, and more if you are going to program ValCalc.

1.10 eval ?

Eval

Eval is a little program located in the c-drawer on your boot-partition. It is capable of evaluating expressions including '+', '-', '*' and '/'. It only handles integers.

1.11 Calculator ?

Calculator

Calculator is a tiny version of a desktop calculator with almost no features. It is located in the tools-drawer of your boot-partition.

1.12 Isn't that what eval does ?

CLI Mode

To use ValCalc in CLI Mode simply type ValCalc in a CLI and the expression you want to be evaluate as the first argument and press return. ValCalc will then display the answer or an error message.

This error message will not tell you what the error is. If you can't find the error, use GUI Mode where a more meaningful error message will be displayed.

In CLI Mode ValCalc acts as a replacement for eval, with the improvements that ValCalc handles floating point numbers and have the standard trigonometric functions build-in.

1.13 Hmm, I wonder what this button does...

GUI Mode

To use ValCalc in GUI Mode double-click on it's icon or type ValCalc, without any arguments, in a CLI. When ValCalc is runned in this mode it opens its main window, with an amazing similarity to a desktop-calculator, on the Workbench screen.

Using the main window	How to operate the main window.
Using the menus	What is in the menus and how do I use them.
Variables	A short explanation of variables.
Bases	A short explanation of bases.
Programming ValCalc	Now this is the fun part.

1.14 That looks nice

The Main Window

The main window is used for entering expressions, and seeing the results.

Expression are entered in three ways:

- 1) Using the bottoms in the bottom of the window.
- 2) Typing the expression directly using the keyboard.
- 3) Clicking on a previously entered expression which are shown in the top of the window.

Expressions are evaluated when the Return key or the enter button are pressed. While entering the expression, it is showed in the text-field in the middle of the window.

1.15 Let's see, where can I find Quit

The Menues

Here is what is in the menues:

Project

About Shows some information about ValCalc
Quit Quits the program

Windows

Calculator Brings the main window to front
Variables Brings up the variables window from where variables can be manipulated.
Bases Brings up the bases window from where bases can be manipulated.
Functions Brings up a window with a listning of the build-in functions. Click on a function to insert it in your expression.

Procedures

Load Procedure Brings up a file-requester. Choose a textfile with a procedure in order to load it.
In case of succes, ValCalc will display a requester saying so. In case of failure, an errormessage stating the type of error and a linenumber, will be displayed in the main window.

Preferences

# of lines	Brings up a number-requester letting the user set how many expressions/results, ValCalc should remember.
# of significant digits	Brings up a number-requester letting the user set an upper limit for how many digits that should be displayed when ValCalc shows a result.
Display mode	This menuitem lets the user choose in which notation answers should be displayed:

	Scientific or Free.
Variables mode	Lets the user set whether variables should be local or global when using procedures
Output to printer	Lets the user set what should be echoed to the printer: None: Nothing gets echoed to the printer. Partial: Everything showed in the top of the main window gets echoed to the printer. ← Full: Same as partial plus requesters with messages to the user also gets echoed.

1.16 "

Scientific notation

In this notation numbers are displayed in the following way:

```
120 as 1.20e2
0.3 as 3.0e-1
1 as 1.0e0
...
```

1.17 "

Free notation

This is the standard notation. If the numbers contains ten or more ciffers, the scientific notation is used.

1.18 "

Local variables

In this mode a procedure will get its own copy of the variables. This means that the procedure will know all variables defined before the procedure was entered, but changes in existing variable, and declaration of new ones, will not affect variables outside the procedure.

1.19 "

Global variables

In this mode changing of variables and declaration of new ones will be remembered after the procedure returns.

1.20 Nothing ever stays the same

Variables window

This window shows a listing of existing variables. When you select one (eg. click on the name), the two buttons at the bottom of the window becomes 'click-able'. 'Delete' deletes the selected variable. 'Rename' prompts you for a new name and renames the variable unless a variable already has the new name.

Double-clicking on a variable inserts it in the expression in the main window. To add a new variable enter the following expression in the main window:

```
<variabelname>=<value>
```

Close the window by clicking in the close-gadget.

1.21 It's a long shot and Mantle runs for the third base...

Bases window

If you haven't read the Bases-section, you should do that before continuing here.

The window shows a listing of existing bases with 'default' at the current default base.

At the bottom of the window is four buttons:

- 1) Delete Select a base and press this button to delete it. The default base cannot be deleted.
- 2) Rename Select a base and press this button to change the base identifier.
- 3) Default Select a base and press this button to make it the default base.
- 4) New Press this button to make a new base. ValCalc will prompt you for identifier og value.

1.22 The values they are a-changing

Variables

A variable is a symbolic name you can assign a numeric value and then use the name instead of the numeric value. Example:

```
pi = 3.14
```

```
r = 3.0
```

```
o = pi*r
```

You can change a variables value by assigning it a new value (hence the name 'variable').

A variable must be assigned a value before it is used.

1.23 First we must have a base

Bases

If you are familiar with the concept 'base' skip this paragraph.

To explain what a base is lets take an example:

The number 123 could also be written as:

$1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$

Notice the 10's, they are called the basevalue. Sometimes it is easier to use another basevalue, for example 16. Then 123 would mean:

$1 \cdot 16^2 + 2 \cdot 16^1 + 3 \cdot 16^0$

In ValCalc a base consist of a basevalue and a base identifier. The basevalue is explained above. The base indentifier is a letter that must be put after a number in order to use the coherent basevalue. If no baseidentifier is put after a number it is considered as using the default base. For setting the default base, use the bases window.

From the begining four bases are defined:

Decimal	identifier: d	basevalue: 10	(default)
Hexadecimal	identifier: h	basevalue: 16	
Octal	identifier: o	basevalue: 8	
Binary	identifier: b	basevalue: 2	

When you are using basevalues above 10 letters in uppercase must be used.

1.24 ...and when Nina saw this, she got a funny expression on her face

Expressions

The input given to ValCalc are called expressions.

For the ones who know BNF there is one for expression right here.

If you do not know BNF a more ad hoc explanation comes here.

Basically an expression consists of operators with numbers or functions between.

The functions `sqr()`, `sin()`, `cos()`, `tan()`, `asin()`, `acos()`, `atan()`, `sinh()`, `cosh()`, `tanh()`, `sech()`, `csch()`, `exp()`, `ln()`, `log()`.

The operators is: `+`, `-`, `*`, `/`, `<`, `>`, `<=`, `>=`, `==` and `!=`. In addition there is the faculty operator `'!'` that only has a number/function in front of the operator, eg. `'3!'` or `tan(8)!`.

Finally we have `'('` and `')'`.

Let us take an example:

`sin(2)+3*7-5!/5+4*(log(4*4.5)+6)`

For numbers the scientific notation can be used:

`1.7e-3` or `4.6e+5` and finally `6.3e5`

Variables can be used instead of numbers.

1.25 Express yourself

BNF for expressions

expression ::=

term '+' expression |
term '-' expression |
term '>' expression |
term '<' expression |
term '<=' expression |
term '>=' expression |
term '==' expression |
term '!=' expression |
term

term ::=

primary2 '/' term |
primary2 '*' term |
primary2

primary2 ::=

term '^' primary2 |
primary3

primary3 ::=

primary2 '!' |
primary

primary ::=

number |
name [= expression] |
'-' expression |
'(' expression ')' |
'Sqr(' expression ')' |
'Sin(' expression ')' |
'Cos(' expression ')' |
'Tan(' expression ')' |
'Asin(' expression ')' |
'Acos(' expression ')' |
'Atan(' expression ')' |
'Sinh(' expression ')' |
'Cosh(' expression ')' |
'Tanh(' expression ')' |
'Sech(' expression ')' |
'Csch(' expression ')' |
'Exp(' expression ')' |
'Ln(' expression ')' |
'Log(' expression ')'

number ::=

ciffers[.ciffers][e[+|-]ciffers] |
[ciffers].ciffers[e[+|-]ciffers]

ciffers ::=

ciffer |
ciffer ciffers

ciffer ::= '0' | '1' | '2' | ... | '9' | 'A' | 'B' | 'C' | ... | 'Z'

```

name ::=
    letter |
    letter name

letter ::= 'a' | 'b' | 'c' | ... | 'z' | 'A' | 'B' | 'C' | ... | 'Z'

```

1.26 Let us try to follow the correct procedure

Procedures

Procedures can be used to program ValCalc by adding new functions or replacing the build-in functions. To make a function do the following:

- 1) Start a text-editor
- 2) Write the procedure (the syntax will be described below)
- 3) Save the procedure in a file
- 4) Select 'Load procedure' from the Procedures-menu and select the file
- 5) If there was no errors in the procedure, it can now be used in expressions

If you make a procedure with same name as an existing function or procedure, it will replace the old one.

Syntax

The language used by ValCalc is a subset of 'C', so if you know 'C', a look at this example should be explanation enough.

For the ones who know BNF there is one for procedures right here.

If you do not know BNF I will explain the syntax in words here.

A procedure must always have a name and contain at least one statement, the last statement must, with some exceptions, be a `return()` statement:

```

Myproc()
{
    return(1);
}

```

The only thing that Myproc does, is returning 1. This means that writing 'Myproc()' in a expression will be the same as writing '1'. Procedures are able to take input, referred to as 'arguments', an example:

```

Plus(x,y)
{
    return(x+y);
}

```

Plus returns the sum of its two arguments, making `Plus(3,4)` in an expression equivalent to '3+4'.

The only statement used so far is 'return', there are three more: while, if-else and assign.

Let us take an example containing them all:

```
AnotherProc(x,y,z)
{
    temp1 = z;
    temp2 = z;
    while(temp1 > 0)
    {
        z = z+x+y;
        temp1 = temp1-1;
    }
    if(z > temp2) return(temp2);
    else return(z);
}
```

The first two lines are assignment-statements; the value of *z* are assigned to two new variables, *temp1* and *temp2*. Next comes a while-statement.

Translated to english the meaning is: So long as '*temp1* > 0' is true do the statements embraced by the '{' and the '}'. If only one statement is to be done, the '{' and '}' are not needed.

The two assign-statements in the while-statement is a bit spurious since the same variable stands on both sides of the '='. The semantic here is: evaluate the expression on the right side and then assign the variable on the left side this value.

Finally we have an if-else statement. Again translated to english the meaning is: If '*z* > *temp2*' is true do '*return(temp2);*' else do '*return(z);*'. More than one statement may follow '*if(z > temp2)*' or '*else*' if they are embraced by '{' and '}'. The else-statement is optional.

Notice that the last statement is not a return-statement. That is because no matter what the values of *z* and *temp2* are, a return-statement will be reached.

Comments

There are two ways in ValCalc to make comments:

```
p()
{
    // This is a comment, all text from the '//' to end-of-line will be ignored
    /* This is another comment, all text here be ignored */
    return(1);
}
```

1.27 "

BNF for procedure

```
procedure ::=
    name '(' [arguments] ')' {' statement_block '}'
```

```
arguments ::=
    name |
    name ',' arguments
```

```
statement_block ::=
```

```

statement |
statement statement_block

statement ::=
  name '=' expression ';' |
  'while(' expression ')' statements |
  'if(' expression ')' statements [ 'else' statements ] |
  'return(' expression ');'

statements ::=
  statement |
  '{' statement_block '}'

```

For a BNF-notation of 'expression' and 'name' see bnf for expression

1.28 "

Example of a procedure

```

p(x,y,z) // Just a little comment
{
  if(z == 1)
    return(x+y);
  if(z == 2)
    return(x-y);
  if(z == 3)
  {
    if(x < y) return(x);
    else return(y);
  }
  if(z == 4) /* Recursive version of fac */
  {
    if(x <= 1) return(1);
    else return(x * p(x-1,y,z));
  }
  if(z == 5) /* Iterative version of fac */
  {
    temp = 1;
    while(x > 1)
    {
      temp = temp * x;
      x = x-1;
    }
    return(temp);
  }
  return(z);
}

```

1.29 "

Todo

Below is a list of things I am thinking about adding to ValCalc. I am not saying that they will be added though.

- o Make ValCalc an Arexx command host (Anyone having some C/C++-code showing how ↵?)
- o 'Functions' to make it possible to do input/output in procedures/expressions
- o Make the preferences saveable.
- o Typed variables, eg. integers, chars, arrays etc.
- o Clipboard support
- o Make this documentation on-line
- o Localization
- o Any good idea I come up with (if any :-)

1.30 VALstory

1.0 First public release

1.31 ...without whom this would all have been possible...

Acknowledgements

The initial userinterface was made with GadToolsBox by Jan Van Den Baard.
All requesters are made with ReqTools by Nic Francois.
The font being used, XHelvetica, is from MagicWB.
The icons was drawn by Carsten Rudi Hess.