scalc Documentation

Christopher Olah

September 7, 2008

scalc is a C++ implimented scientific calculator with many non-standard functions and provided variables. It has some programming-language-like functionality. It can operate either in an interactive shell or as a stream calculator.

Contents

1 Liscense **1**

2 Installation **1**

3 Invocation and Input **2**

4 Arithmatic **2**

5 As a Programming Language **3**
5.1 Variable Assignment 3
5.2 Control Structure 4

1 Liscense

scalc is released under the GNU GPL. All it's documentation is released under the GNU FDL.

2 Installation

scalc is written for Unix-like operating systems. It has only been tested on Linux. It has only been compiled on the GNU Compiler for C++ but uses no GNU specific features (asm, likely, et cetera) and should be able to compile on any ISO C++ compliant compiler. It depends only on standard libraries and

it's own.

To install, compile the libraries in the following order: fstring.h, science_math.h, solver.h, scalc.cpp. Name the output of compiling scalc.cpp whatever you want to type to call scalc. (For your convenience, a Makefile is included to do this (output name scalc). Just run make.) Move the resulting binary into your PATH.

A .deb file is provided for users of i386 systems. Just run dpkg –install scalc-i386-0.8.deb.

3 Invocation and Input

scalc may be invoced without argument to create an interactive shell. In this mode, simply type in an expression and press 'enter'.

scalc may also be invoked as a stream calculator with a unspaced argument or quoted argument. The argument is an equation that will be applied on all streams which are evaluated before this point and represented by the variable 's' or 'S'.

4 Arithmatic

The expression will be evaluated based on simple rules. Firstly, anything in brackets (marked by '(' and ')') wil be evaluated prior to anything else. Then it evaluates comperatives ('==' (comperative equals) or '!=' (comperative doesn't equal)). (Technically, special goes here, but we'll discuss it later.) Next, exponents are evaluated (a to the power of by is a^b). The multiplication('*') and division ('/') are performed. Finally, addition ('+') and subtraction ('-') are applied.

As mentioned previously, there is a stage called 'special' before exponents. It has to do with the actual parsing of a number. When parsing, the program doesn't care if a 'number' is 21 or 2!. All it cares about is that it doesn't contain '^','*','/','-', or '+' (or '','', ';', '"', or '='... later). So this leaves room for other things to implimented and is where the majority of functions and variables are. Special is implemented as a recursive series of 'else if's and it is difficult to remember what will be evaluated first, so one should *always* use brackets.

Representation	Function/Value	Representation	Function/Value
x	Number	g	g
mx	Mass of Element	G	G
AC	Avagadro's Constant	c	c
LperM	Literes per Mole of gass at STP	e_Q	Charge of Electron
R	Ideal Ratio	e	e
sTp/StP	Standard Temperature	KMHtoMS	KM/H to M/S
stP/STp	Standard Preasuer	MStoKMH	M/S to KM/H
sin$x$	sine	CtoK	Celcius to Kelvin
cos$x$	cosine	KtoC	Kelvin to Celcius
tan$x$	tangent	$x\&\&y$	AND
asin$x$	archsine	$x\|\|y$	OR
acos$x$	archcosine	$!x$	NOT
atan$x$	archtangent	$x!$	Factorial (ERROR)
log$x$	logarithm base 10	dx	dice of x sides
sqrt$x$	squareroot	(Fx)	
p/pi	Pi	(s/S)	
(var)		x@y	x interpreted by symbols y

5 As a Programming Language

While originally intended to be a calculator, scalc has gained many of the properties of a programing language. These are still largely in development.

5.1 Variable Assignment

There are four assignable variables: out, var, F(x), and @. out is output, var is a user defined variable, F(x) is a mathamatical funtion with all instances of x in it being replaced by the value it is called with, and @ is default number parsing device. They are all string-like except var which is double-like.

Variables are assigned by the equals sign. To assign a string, put it in quotations (otherwise the interpretter will try to evaluate it). End the assignment with a semicolon.

For example, the Hello World program would be: out="Hello_World!";

What happens in normal arithmatic? If a statment doens't end with a semicolon it is prepended with "out=". (So Hello World could be done as "Hell_World!")

5.2 Control Structure

Control stuructures consist of a name and arguent and data.

Name	Execution
if	If argument is true, execute data.
repeat	Execute data argument times.
while	While argument is true, execute data in loop.