NAME

rpntutorial − Reading RRDtool RPN Expressions by Steve Rader

DESCRIPTION

This tutorial should help you get to grips with RRDtool RPN expressions as seen in CDEF arguments of RRDtool graph.

Reading Comparison Operators

The LT, LE, GT, GE and EQ RPN logic operators are not as tricky as they appear. These operators act on the two values on the stack preceding them (to the left). Read these two values on the stack from left to right inserting the operator in the middle. If the resulting statement is true, then replace the three values from the stack with ‘‘1’’. If the statement if false, replace the three values with ‘‘0’’.

For example, think about ‘‘2,1,GT’’. This RPN expression could be read as ‘‘is two greater than one?’’ The answer to that question is ‘‘true’’. So the three values should be replaced with ‘‘1’’. Thus the RPN expression 2,1,GT evaluates to 1.

Now consider ‘‘2,1,LE’’. This RPN expression could be read as ‘‘is two less than or equal to one?’’. The natural response is ‘‘no’’ and thus the RPN expression 2,1,LE evaluates to 0.

Reading the IF Operator

The IF RPN logic operator can be straightforward also. The key to reading IF operators is to understand that the condition part of the traditional ‘‘if X than Y else Z’’ notation has *already* been evaluated. So the IF operator acts on only one value on the stack: the third value to the left of the IF value. The second value to the left of the IF corresponds to the true (‘‘Y’’) branch. And the first value to the left of the IF corresponds to the false (‘‘Z’’) branch. Read the RPN expression ‘‘X,Y,Z,IF’’ from left to right like so: ‘‘if X then Y else Z’’.

For example, consider ‘‘1,10,100,IF’’. It looks bizarre to me. But when I read ‘‘if 1 then 10 else 100’’ it’s crystal clear: 1 is true so the answer is 10. Note that only zero is false; all other values are true. ‘‘2,20,200,IF’’ (‘‘if 2 then 20 else 200’’) evaluates to 20. And ‘‘0,1,2,IF’’ ("if 0 then 1 else 2) evaluates to 2.

Notice that none of the above examples really simulate the whole ‘‘if X then Y else Z’’ statement. This is because computer programmers read this statement as ‘‘if Some Condition then Y else Z’’. So it’s important to be able to read IF operators along with the LT, LE, GT, GE and EQ operators.

Some Examples

While compound expressions can look overly complex, they can be considered elegantly simple. To quickly comprehend RPN expressions, you must know the algorithm for evaluating RPN expressions: iterate searches from the left to the right looking for an operator. When it’s found, apply that operator by popping the operator and some number of values (and by definition, not operators) off the stack.

For example, the stack ‘‘1,2,3,+,+’’ gets ‘‘2,3,+’’ evaluated (as ‘‘2+3’’) during the first iteration and is replaced by 5. This results in the stack ‘‘1,5,+’’. Finally, ‘‘1,5,+’’ is evaluated resulting in the answer 6. For convenience, it’s useful to write this set of operations as:

```
1) 1,2,3,+,+    eval is 2,3,+ = 5    result is 1,5,+
2) 1,5,+        eval is 1,5,+ = 6    result is 6
3) 6
```

Let’s use that notation to conveniently solve some complex RPN expressions with multiple logic operators:

```
1) 20,10,GT,10,20,IF  eval is 20,10,GT = 1     result is 1,10,20,IF
```

read the eval as pop ‘‘20 is greater than 10’’ so push 1

```
2) 1,10,20,IF         eval is 1,10,20,IF = 10  result is 10
```

read pop ‘‘if 1 then 10 else 20’’ so push 10. Only 10 is left so 10 is the answer.

Let’s read a complex RPN expression that also has the traditional multiplication operator:

```
1) 128,8,*,7000,GT,7000,128,8,*,IF  eval 128,8,*       result is 1024
2) 1024   ,7000,GT,7000,128,8,*,IF  eval 1024,7000,GT  result is 0
3) 0,             7000,128,8,*,IF  eval 128,8,*       result is 1024
4) 0,             7000,1024,   IF                     result is 1024
```

Now let's go back to the first example of multiple logic operators, but replace the value 20 with the variable ''input'':

```
1) input,10,GT,10,input,IF  eval is input,10,GT  ( lets call this A )
```

Read eval as ''if input > 10 then true'' and replace ''input,10,GT'' with ''A'':

```
2) A,10,input,IF            eval is A,10,input,IF
```

read ''if A then 10 else input''. Now replace A with it's verbose description again and — voila!−−you have an easily readable description of the expression:

```
if input > 10 then 10 else input
```

Finally, let's go back to the first most complex example and replace the value 128 with ''input'':

```
1) input,8,*,7000,GT,7000,input,8,*,IF  eval input,8,*     result is A
```

where A is ''input * 8''

```
2) A,7000,GT,7000,input,8,*,IF          eval is A,7000,GT  result is B
```

where B is ''if ((input * 8) > 7000) then true''

```
3) B,7000,input,8,*,IF                  eval is input,8,*  result is C
```

where C is ''input * 8''

```
4) B,7000,C,IF
```

At last we have a readable decoding of the complex RPN expression with a variable:

```
if ((input * 8) > 7000) then 7000 else (input * 8)
```

Exercises

Exercise 1:

Compute ''3,2,*,1,+ and ''3,2,1,+,*" by hand. Rewrite them in traditional notation. Explain why they have different answers.

Answer 1:

```
3*2+1 = 7 and 3*(2+1) = 9.  These expressions have
different answers because the altering of the plus and
times operators alter the order of their evaluation.
```

Exercise 2:

One may be tempted to shorten the expression

```
input,8,*,56000,GT,56000,input,*,8,IF
```

by removing the redundant use of ''input,8,*'' like so:

```
input,56000,GT,56000,input,IF,8,*
```

Use traditional notation to show these expressions are not the same. Write an expression that's equivalent to the first expression, but uses the LE and DIV operators.

Answer 2:

```
if (input <= 56000/8 ) { input*8 } else { 56000 }
input,56000,8,DIV,LE,input,8,*,56000,IF
```

Exercise 3:

Briefly explain why traditional mathematic notation requires the use of parentheses. Explain why RPN

notation does not require the use of parentheses.

Answer 3:

```
Traditional mathematic expressions are evaluated by
doing multiplication and division first, then addition and
subtraction.  Parentheses are used to force the evaluation of
addition before multiplication (etc).  RPN does not require
parentheses because the ordering of objects on the stack
can force the evaluation of addition before multiplication.
```

Exercise 4:

Explain why it was desirable for the RRDtool developers to implement RPN notation instead of traditional mathematical notation.

Answer 4:

```
The algorithm that implements traditional mathematical
notation is more complex then algorithm used for RPN.
So implementing RPN allowed Tobias Oetiker to write less
code!  (The code is also less complex and therefore less
likely to have bugs.)
```

AUTHOR

Steve Rader <rader@wiscnet.net>