Leone Learning Systems, Inc.
Wonder. Create. Grow.

Leone Learning Systems, Inc.
237 Custer Ave
Evanston, IL 60202
Email tj@leonelearningsystems.com

Phone 847 951 0127
Fax 847 733 8812

Infix Operations and Order of Evaluation

TJ Leone

October 2004

Acknowledgements

Special thanks to Justin Eberlein for being a careful reader and an excellent student. This document has fewer mistakes because of him.

Introduction

Until now, we have used the operations `sum`, `difference`, `product`, and `quotient` to carry out the arithmetic operations of addition, subtraction, multiplication and division. In this unit, we will learn to use different operations called *infix operations*. Here is a table to show you the infix operations that correspond to `sum`, `difference`, `product`, and `quotient`:

`sum 7 8`	`7 + 8`
`difference 9 3`	`9 - 3`
`product 5 4`	`5 * 4`
`quotient 12 6`	`12 / 6`

Fortunately, standard keyboards come with + and – signs. Since there isn't a $\times$ on a standard keyboard, we use * for multiplication. Since there isn't a $\div$ on the keyboard, we use /.

Infix operations are nice because they look like the operations we usually use when we write down math problems with a paper and pencil. However, there are special rules for using infix operations that you need to learn so that you can use them correctly. We will discuss these rules in this unit.

Standard order of evaluation

Before we discuss the order of evaluation for infix operations, let's review the order of evaluation for typical Logo instructions by looking at an example we've seen before:

```
print sum 4 product 10 2
```

Logo looks at the first word in the message, which is `print`. Logo knows that `print` is the name of a procedure and that `print` needs one input, so Logo continues reading to find the input. The next word that Logo sees is `sum`. Logo knows that this is also the name of a procedure and that `sum` needs two inputs, so Logo continues reading to find the inputs for `sum`. The first word after sum is 4. Logo knows that 4 is a constant. In other words, 4 always evaluates to 4 (it's not a variable or an operation that can evaluate to something else), so Logo assigns 4 as the first input to `sum` and goes looking for the second input to `sum`. The next word after 4 is `product`, which is an operation that requires two inputs. Logo reads further and finds two inputs for `product`, 10 and 2.

Once `product` receives the two inputs it needs, it can produce its output. In this case, since its inputs are 10 and 2, its output is 20. This output becomes the second input to `sum`. Now `sum` has the two inputs it needs, 4 and 20, and it produces its output 24. This output becomes the input for `print`, which prints out the value 24.

The operations `sum` and `product` are called *prefix operations*, because the names of the operations are written before the inputs of the operation.

Special operations and infix evaluation

As you saw in the introduction, the creators of Logo provide alternative ways to write arithmetic instructions.

For example, instead of

```
print sum 2 3
```

You can write

```
print 2 + 3
```

Try it.

The + operation is called an *infix operation* because it comes *in between* its two inputs.

Because we understand how prefix operations work, we can understand the difference between these instructions:

```
print sum 2 product 3 4
print product sum 2 3 4
```

Notice that in both of these instructions, the numbers 2, 3 and 4 are in the same order (2 comes first, then 3, then 4). What changes is the *order of the operations* `sum` and `product`.

Since the infix operations were modeled after the arithmetic operations used on pencil and paper, they follow the rules used by mathematicians in evaluating expressions with pencil and paper. These rules are called *precedence rules*.

Precedence rules: Addition and Subtraction

Why do we need precedence rules? Consider this instruction:

```
print 2 + 3 + 4
```

Keeping the numbers in order (2, then 3, then 4) there are two ways we might write this instruction in infix form:

```
print sum sum 2 3 4
print sum 2 sum 3 4
```

In the first case, we start by adding the 2 and the 3, and then add the 4. In the second case, we start by adding the 3 and 4, and then add the 2. In both cases, we get the same answer (what is it?), so order of operations doesn't matter much here.

How about this one?

```
print 8 - 5 - 2
```

```
print difference difference 8 5 2
print difference 8 difference 5 2
```

Try evaluating each of these. What results do you get?

In the first case, we start by subtracting the 5 from the 8, and then subtract 2 from the result. In the second case, we start by subtracting the 5 from the 2, and then subtract 8 from the result.

Since we get two different answers for the two different cases, we see that order of operation is important for subtraction. So we need a precedence rule to decide the order of subtractions. The rule is, in cases like the one above, we *subtract from left to right*. In other words, the problem above is solved like this:

```
8 - 5 – 2 = 3 – 2 = 1
```

so

```
print 8 - 5 - 2
```

is equivalent to

```
print difference difference 8 5 2
```

Precedence rules: Addition and Multiplication

How about this instruction?

```
print 2 + 3 * 4
```

Which prefix version is its equivalent? Or are both the same?

```
print product sum 2 3 4
print sum 2 product 3 4
```

Try all of these instructions before you go on.

The precedence rule for this example is *do multiplication before addition*. In other words, if you are using pencil and paper, the expression is solved like this:

```
2 + 3 * 4 = 2 + 12 = 14
```

The prefix instruction that is equivalent to

```
print 2 + 3 * 4
```

is

```
print sum 2 product 3 4
```

In general, addition and subtraction are considered to be on the same "level" and multiplication and division are on the next higher "level", so multiplication and division are done before addition and subtraction.

Precedence rules: Parentheses

We saw that

```
print 8 - 5 - 2
```

is equivalent to

```
print difference difference 8 5 2
```

and

```
print 2 + 3 * 4
```

is equivalent to

```
print sum 2 product 3 4
```

But what if we want to use infix operations to write the equivalents of the following?

```
print difference 8 difference 5 2
print product sum 2 3 4
```

Standard algebra and Logo give us a way to override other precedence with a new precedence rule: *do operations in parentheses first*. Here's how to write the two instructions above using infix operations:

```
print 8 – (5 – 2)
print (2 + 3) * 4
```

Solved Problems

1. *Problem:* Simplify the following expressions with paper and pencil:

   a. $8 – 2 * 3 – 1$
   b. $12 / (6 – 2)$
   c. $(4 + 3) * (8 – 5)$

   Solution:

   a. Since multiplication comes before addition or subtraction, we multiply the 2 and 3 together before we do anything else:

      $8 – 2 * 3 – 1 = 8 – 6 – 1$

      Now, since we only have subtractions left, we subtract from left to right:

      $8 – 6 – 1 = 2 – 1 = 1$

   b. Since operations in parentheses come first, we have

      $12 / (6 – 2) = 12 / 4$

      Then, we just do the division:

      $12 / 4 = 3$

   c. Since operations in parentheses come first, we have

      $(4 + 3) * (8 – 5) = 7 * 3$

      Doing the multiplication gives us

      $7 * 3 = 21$

Solved Problems (continued)

2. *Problem:* Convert the following prefix instructions to infix instructions. Test out the instructions to make sure you get the same answers for the prefix and infix versions.

   a. `print difference 10 product 4 2`
   b. `print sum product 2 2 7`
   c. `print product 2 sum 2 7`

   Solution:

   a. As Logo evaluates this instruction, the first operation to occur will be the multiplication of 4 and 2, which provides the second input to difference. So the infix form is:

      ```
      print 10 – 4 * 2
      ```

      we don't need parentheses because the precedence rules tell us that multiplication is done before subtraction. However, it would still be correct to write:

      ```
      print 10 – (4 * 2)
      ```

   b. In this instruction, Logo evaluates `product 2 2` as the first input of `sum`. This means multiplication before addition, so again, we don't need parentheses:

      ```
      print 2 * 2 + 7
      ```

      with (unnecessary but still correct) parentheses, the instruction would be:

      ```
      print (2 * 2) + 7
      ```

   c. This instruction does addition before multiplication, so *we must have parentheses*:

      ```
      print 2 * (2 + 7)
      ```

Solved Problems (continued)

3. *Problem:* Convert the following infix instructions to prefix instructions. Test out the instructions to make sure you get the same answers for the infix and prefix versions.

   a. `print 2 * 4 – 3`
   b. `print 2 * (4 – 3)`
   c. `print (3 + 2 * 6) / (7 – 2)`

   Solution:

   a. Since multiplication is done before subtraction, Logo must evaluate `product 2 4` before the subtraction is done. So we end up with this:

      ```
      print difference product 2 4 3
      ```

   b. In this case, the parentheses require that the subtraction is done before the multiplication, so Logo needs to evaluate `difference 4 3` before the product is evaluated:

      ```
      print product 2 difference 4 3
      ```

   c. Because of the parentheses, we can see that the last operation we want to do is the division. Notice that the last prefix operation evaluated by Logo is the first operation on the instruction line. So we start with this:

      ```
      print quotient ? ?
      ```

      Now we need to figure out the two inputs to `quotient`. We get the first input from the infix expression `(3 + 2 * 6)` and the second input from the infix expression `(7 – 2)`

      ```
      print quotient sum 3 product 2 6 difference 7 2
      ```

Supplementary Problems

1. *Problem:* Simplify the following expressions with paper and pencil:

   a. 1 + 2 * 6 – 4
   b. 10 / (7 – 2)
   c. (5 – 3) * (8 – 5)

   Solution:

   a. 1 + 2 * 6 – 4 = 1 + 12 – 4
   b. 10 / (7 – 2) = 10 / 5 = 2
   c. (5 – 3) * (8 – 5) = 2 * 3 = 6

2. *Problem:* Convert the following prefix instructions to infix instructions. Test out the instructions to make sure you get the same answers for the prefix and infix versions.

   a. `print sum 3 product 6 5`
   b. `print difference quotient 24 12 2`
   c. `print difference 24 quotient 12 2`

   Solution:

   a. `print 3 + 6 * 5`
   b. `print 24 / 12 – 2`
   c. `print 24 – 12 / 2`

3. *Problem:* Convert the following infix instructions to prefix instructions. Test out the instructions to make sure you get the same answers for the infix and prefix versions.

   a. `print 5 * 7 – 3`
   b. `print 5 * (7 – 3)`
   c. `print (3 + 6) / (7 – 2 * 3)`

   Solution:

   a. `print difference product 5 7 3`
   b. `print product 5 difference 7 3`
   c. `print quotient sum 3 6 difference 7 product 2 3`

The Author

TJ Leone owns and operates Leone Learning Systems, Inc., a private corporation that offers tutoring and educational software. He has a BA in Math and an MS in Computer Science, both from the City College of New York. He spent two years in graduate studies in education and computer science at Northwestern University, and six years developing educational software there. He is a former Montessori teacher and currently teaches gifted children on a part time basis at the Center for Talent Development at Northwestern University in addition to his tutoring and software development work. His web site is http://www.leonelearningsystems.com.