Wednesday January 23

Testing the short projects
Review: arrays and iteration

Tests

I didn’t give any instructions for testing the three short programming projects
Some suggestions:
1. for sum.rb, just pick a small set of numbers, e.g.
   % sum.rb 5 6 7 8
   26
   6.5

   Note: what do you suppose will happen if the list has no numbers?
   % sum.rb
   0
   NaN
   
   An nice extra credit opportunity: have your program test for a list of size 0

Tests (cont’d)

2. to test the “grep” program you can use any text file -- including a Ruby program
   % grep.rb ne < grep.rb
   while line = STDIN.gets
     puts line
   % grep.rb foo < grep.rb

   Note: I forgot to mention a slight bit of Unix weirdness for this program
   If your program gets input both from the command line (the pattern, in this case) and
   from the standard input, you need to put STDIN in front of gets:
   while line = STDIN.gets ...
   (more on this later in the term)

Project Due Jan 28

Don’t forget the project is due Monday
✦ send e-mail to conery@cs.uoregon.edu by 5:00 PM
✦ Try to work on the programs before Friday so you can ask questions in class...
Main Topics from Jan 14-16

- The process of performing an action several times is known as **iteration**
  - evaluate an expression for every element in a container, using a special method known as an **iterator**
  - `a.each { |x| .... }` or `a.each do |x| .... end`
  - the code between `{` and `}` or between `do` and `end` is a **block**
- repeat an expression a fixed number of times
  - `n.times { |i| .... }`
- repeat a set of statements until a specified condition is met
  - `while expr
      ...
    end`

Main Topics from Jan 14-16 (cont’d)

- Often we want to evaluate an expression only when certain conditions are met
  - The simplest form of conditional evaluation uses a **modifier** attached to the end of a statement
  - Example: counting the nonzero elements in an array
    ```ruby
    a = [10, 9, 0, 10, 7, 0, 9]
    n = 0
    a.each do |x|
      n += 1 if x > 0
    end
    n
    #=> 5
    ```

Main Topics from Jan 14-16 (cont’d)

- To determine whether a condition is true in a while statement or a conditional expression Ruby evaluates a boolean expression
  - the value of the expression is true or false
    - `true` and `false` are special objects in Ruby
  - Any object can be used in a while statement or an if modifier
    - `false` is false
    - `nil` is false
    - anything else is true
  - A common construction:
    ```ruby
    while line = gets ...
      #gets returns a String object or nil
      #the loop executes as long as there are strings in the input stream
    end
    ```

Today’s Topic

- Continuing where we left off in talking about conditional expressions
  - A new type of container known as an “associative array” or “hash”