Levi George

03/27/2021

Prof. Amal Khalifa

CS 350: Programming Language Design

Homework 5: Ruby and Sub-Programs

i. What is the syntax for declaring parameters with default values?

a.
```ruby
def myFunc(myDefParam = 1, myDefParam2 = 3)
  #stuff happens here
end
```

**b. How are the default values used then the function is called?**

c. The default values are used only when arguments are not passed. However, we must pass arguments in the proper order when we declare default parameters. Otherwise, we experience undefined behavior or logic errors.

ii. How can functions be passed as arguments?

a. You would first declare a function, and write the program to use the function. However, when you pass it, you must pass it as a symbol.

b.
```ruby
def printFunc
  puts "Print Function was Activated"
end

def funcActivator(fun)
  method(fun).call
end

funcActivator(:printFunc)
```
```
bundle exec ruby main.rb
Print Function was Activated
```

iii. Define a simple recursive function and trace its call stack.

a.
```ruby
def recursingFunc(string)

  for i in string
    putc(i)
  end

   puts(" ")

  if(string.empty?)
   return
  end

  recursingFunc(string[1..-1])

end

recursionArr = ['C', 'P', 'U']

recursingFunc(recursionArr)
```
```
bundle exec ruby main.rb
CPU
PU
U
```

b.

```
Call 1

print "CPU"

Is print empty? (no)
call recursingFunc(send "PU")

->Call 2
  |
  |print "PU"
  |
  |Is print empty? (no)
  |call recursingFunc(send "U")
  |
--->Call 3
    |
    |print "U"
    |
    |Is print empty? (no)
    |call recursingFunc(send "")
    |
------->Call 4
        |
        |print ""
        |
        |Is print empty? (yes)
        |
        |return
        |
-------|Call 4 END
   ||
---|Call 3 END
  |
-|Call 2 END

Call 1 END
```

iv. Is it possible to pass a variable number of parameters to a method? How?

a. Yes, we use the *more keyword as an argument

b.

```
def longVarList(*more)
  puts more
end

longVarList(1,2,3,4,5,6,7,8)

longVarList('a', 'b', 'c')
```

```
bundle exec ruby main.rb
1
2
3
4
5
6
7
8
a
b
c
```

v. Can subprograms be overloaded? Why?

    a. Overloading is not directly supported, you could use different argument list lengths to implement some kind but not true overloading. This is because Ruby is dynamically typed, while overloading is only possible in statically typed languages.

vi. Is there any support for generic subprograms? Why?
    a. No, for much the same reason that overloading is not supported.