Article
Danny Wijnschenk · Nov 21, 2017 5m read

Advent of Code 2016 Day21: Scrambled Letters and Hash

This is a series of programming challenges for beginners and experienced Caché programmers.

For an introduction : go to article https://community.intersystems.com/post/advent-code-2016-day1-no-time-ta...

The challenge of day 21 is about scrambling passwords.

There are a few functions you need to implement that will do operations on a string :

```
- swap position X with position Y
 : means that the letters at indexes X and Y (counting from 0) should be swapped.
- swap letter X with letter Y
  : means that the letters X and Y should be swapped (regardless of where they appear
 in the string).
- rotate left/right X steps
 : means that the whole string should be rotated; for example, one right rotation wou
ld turn abcd into dabc.
- rotate based on position of letter X
 : means that the whole string should be rotated to the right based on the index of l
etter X (counting from 0) as determined before this instruction does any rotations. O
nce the index is determined, rotate the string to the right one time, plus a number o
f times equal to that index, plus one additional time if the index was at least 4.
- reverse positions X through Y
 : means that the span of letters at indexes X through Y (including the letters at X
and Y) should be reversed in order.
- move position X to position Y
 : means that the letter which is at index X should be removed from the string, then
inserted such that it ends up at index Y.
```

Given the instructions in your puzzle input (look for a full explanation to http://adventofcode.com/2016/day/21), what's the scrambled version of the initial string abcdefgh ?

Here is the code i wrote to solve this :

```
Class AOC2016.Day21 Extends AOC2016.Utils
{

ClassMethod Part1(file As %String = "day21.txt", password = "abcdefgh") As %String
{
  #Dim input, instruction as %String
  #Dim iInput, X, Y, temp as %Integer
  #Dim letter as %Char
```

```
  Try {
    Do ..Input(file, .input)
    For iInput=1:1:input {
      Set instruction = $Piece(input(iInput), " ",1,2)
      If instruction="swap position" {
        set X = $Piece(input(iInput)," ",3)+1
        set Y = $Piece(input(iInput)," ",6)+1
        Set temp = $Extract(password,X)
        Set $Extract(password,X)=$Extract(password,Y)
        Set $Extract(password,Y)=temp
      } elseIf instruction="swap letter" {
        set X = $Piece(input(iInput)," ",3)
        set Y = $Piece(input(iInput)," ",6)
        Set password=$Translate(password, X_Y, Y_X)
      } elseIf instruction="move position" {
        Set X = $Piece(input(iInput)," ",3)+1
        Set Y = $Piece(input(iInput)," ",6)+1
        Set letter=$E(password,X)
        Set password=$E(password,1,X-1)_$E(password,X+1,*)
        Set password=$E(password,1,Y-1)_letter_$E(password,Y,*)
      } elseIf instruction="rotate based" {
        Set letter = $Piece(input(iInput)," ",7)
        Set X = $Find(password, letter)-1-1
        Do ..RotateRight(.password, 1+X+$Select(X>3:1,1:0))
      } elseIf instruction="reverse positions" {
        Set X = $Piece(input(iInput)," ",3)+1
        Set Y = $Piece(input(iInput)," ",5)+1
        Set password=$Extract(password,1,X-1)_$Reverse($Extract(password,X,Y))_
$Extract(password,Y+1,*)
      } elseIf instruction="rotate left" {
        Set X = $Piece(input(iInput)," ",3)
        Do ..RotateLeft(.password, X)
      } elseIf instruction="rotate right" {
        Set X = $Piece(input(iInput)," ",3)
        Do ..RotateRight(.password, X)
      }
    }
  } catch {
    Write "Error : ",$ZError,!
  }
  Quit password
}

ClassMethod RotateRight(ByRef str As %String, X)
{
  Set X = X # $Length(str)
  If X > 0 {
    Set str = $E(str_str,$Length(str)+1-X,*-X)
  }
}

ClassMethod RotateLeft(ByRef str As %String, X)
{
  Set X = X # $Length(str)
  If X > 0 {
    Set str = $E(str_str,1+X,$Length(str)+X)
  }
}
```

```
}
```

The second part of the challenge asks to do the reverse : you have to calculate which password was scrambled to end with the scrambled version fbgdceah?

My first thought was to do the reverse of all instructions, starting with the last one.

I could implement all the reverse functions, but there was one instruction - rotate based - that could give more than one possible value.

I did not want to change all my code to a graph based solution, and decided to use brute force : since there are 8 characters in the password, there are 8! combinations that could be the correct starting point.

So I created a method to go over all possible combinations that lead to the scrambled version :

```
ClassMethod Part2()
{
  Do ..Recursive("")
}

ClassMethod Recursive(s) As %String
{
  #Dim ok as %Boolean = 0
  #Dim scrambled as %String
  #Dim iLetter as %Integer
  #Dim letter as %Char

  If $Length(s)=8 {
    Set scrambled = ..Part1(,s)
    If scrambled="fbgdceah" Write !,"found : ",s Set ok=1
  } else {  //try all combinations by adding at each position one of the 8 letters
    For iLetter=1:1:8 {
      Set letter=$E("abcdefgh",iLetter)
      If s'[letter {  //letter must be unique in password
        Set ok = ..Recursive(s_letter)
        If ok Quit
      }
    }
  }
  Quit ok
}
```

Look here for all our solutions so far : https://bitbucket.org/bertsarens/advent2016 and https://github.com/DannyWijnschenk/AdventOfCode2016

Here is the list of all Advent of Code 2016 articles :

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

#Caché #Code Snippet #Contest #ObjectScript