Article
Danny Wijnschenk · Nov 23, 2017 4m read

Advent of Code 2016 Day23: Safe Cracking

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...

Remember the assembunny language we had to code on day 12 (https://community.intersystems.com/post/advent-code-2016-day12-leonardos...)?

There is a new instruction we have to implement which toggles an instruction x lines away (further or back).

- For *one-argument* instructions, inc becomes dec, and all other one-argument instructions become inc.
- For *two-argument* instructions, jnz becomes cpy, and all other two-instructions become jnz.
- The arguments of a toggled instruction are *not affected*.
- If an attempt is made to toggle an instruction outside the program, *nothing happens*.
- If toggling produces an *invalid instruction* (like cpy 1 2) and an attempt is later made to execute that instruction, *skip it instead*.
- If tgl toggles *itself* (for example, if a is 0, tgl a would target itself and become inc a), the resulting instruction is not executed until the next time it is reached

Look for http://adventofcode.com/2016/day/23 for the complete explanation.

So we can re-use much of the code we wrote on day 12.

Basically, I included the new instruction, and made the other instructions more robust (since now, we can have 'illegal' values which we need to ignore).

Here is the code i wrote (it is also worth looking in https://community.intersystems.com/post/advent-code-2016-day12-leonardos...) for the comment that Dmitry wrote about his transpiler of assembunny).

```
Class AOC2016.Day23 Extends AOC2016.Utils
{

ClassMethod Part(file As %String = "day23.txt", part As %Integer = 1)
{
  #Dim iInput, counter, next as %Integer
  #Dim input, instruction, instruction2, param1, param2 as %String
  #Dim var, register, line as %String

  For var="a","b","c","d" Set register(var) = 0
  set register("a") = $Select(part = 1:7,1:12)

  Try {
    Do ..Input(file, .input)
    Set iInput = 1
    set counter=0
    While iInput '> input {
```

```
      set counter=counter+1
      Set line = input(iInput)
      Set instruction=$Piece(line," ",1)
      Set param1=$Piece(line," ",2)
      Set param2=$Piece(line," ",3)
      If instruction = "cpy" {
        If $Data(register(param2)) Set register(param2)=$Select($Data(
register(param1)):register(param1),1:param1)
      } elseIf instruction = "inc" {
        If $Data(register(param1)) Set register(param1) = register(param1) + 1
      } elseIf instruction = "dec" {
        If $Data(register(param1)) Set register(param1) = register(param1) - 1
      } elseIf instruction = "jnz" {
        If $Select($Data(register(param1)):register(param1),1:param1)'=0 {
          Set iInput = iInput + $S($Data(register(param2)):register(param2),1:
param2) -1 //-1 since every loop will do +1 anyway
        }
      } elseif instruction = "tgl" {
        set next=$Select($Data(register(param1)):register(param1),1:param1)+iInput
        If $Data(input(next)) {
          set instruction2=$Piece(input(next)," ",1)
          If instruction2 = "inc" {
            set instruction2="dec"
          } elseIf (instruction2="dec")!(instruction2="tgl") {
            set instruction2="inc"
          } elseIf (instruction2="jnz") {
            set instruction2="cpy"
          } elseIf (instruction2="cpy") {
            set instruction2="jnz"
          }
          set $Piece(input(next)," ",1)=instruction2
        }
      }
      Set iInput = iInput + 1
      //For part 2 : so we can see why it is so much slower
      If (counter#1000000)=0 {
        zwrite register
        Write "#",iInput,!
      }
    }
  } catch {
    Write "Error : ",$ZError,!
  }
  Write "register a = ",register("a"),!
}

}
```

As you can see, I also included the code for part2, which needs another input for register a.

This causes a change in the flow of instructions which makes the whole process much slower. I show the contents of the registers to let you see what is going on.

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