Cubically Documentation

Release 3.1

MD XF

Feb 02, 2018

Contents

1 Quick guide **3**

1.1 What is "Cubically"? . . . 3

1.2 Syntax . . . 3

1.3 Cube commands . . . 3

1.4 Multiple sizes . . . 4

1.5 But how do I interact with the cube? . . . 4

1.6 Cubically sounds fun, but how can I actually use it? . . . 5

1.7 Who's responsible for this mess? . . . 5

- Home
- Code page
- Commands
- Memory
- Functions
- Inner layers
- Syntax

CHAPTER 1

Quick guide

This quick guide will get you started with enough to write some code, but not enough to delve into the pain and terror of the language.

1.1 What is "Cubically"?

Cubically is an esoteric programming language, meaning that it is intentionally difficult to program in.

Other esoteric languages may use a stack or a tape to store data. Cubically, however, uses a built-in Rubik's Cube and three more memory locations with varying levels of read-write restrictions.

Cubically specializes in Rubik's Cubes, so almost all of its commands and functionalities are based around the memory cube. Some, such as XOR and LSHIFT, are really completely useless, but we decided they should be in the language. `¯\_()_/¯`

1.2 Syntax

Digits are arguments and non-digits are commands. A command can only be one letter long. (That rule, which you still need to know, is broken so many times it's not even funny.)

```
code: RL2UD23
does: R       call command R without arguments
       L2     call command L with argument 2
         U    call command U without arguments
          D23 call command D with argument 2, then argument 3 (NOT argument 23)
```

1.3 Cube commands

Any algorithm, or sequence of moves, in Singmaster's notation, is valid Cubically code.

|Command|Description| |-|-| |`R`|rotate the right layer of the cube 90 ° clockwise| |`L`|rotate the left layer of the cube 90 ° clockwise| |`U`|rotate the top layer of the cube 90 ° clockwise| |`D`|rotate the bottom layer of the cube 90 ° clockwise| |`F`|rotate the front layer of the cube 90 ° clockwise| |`B`|rotate the back layer of the cube 90 ° clockwise|

Call any of the above commands with the argument `2` for a 180 ° turn, and the argument `'` or `3` for a counterclockwise turn.

1.4 Multiple sizes

Since v2.0, Cubically has supported more than just a 3x3x3 for the memory cube. In the interpreter's command-line invocation, append an argument to specify the size: `4` for a 4x4x4 memory cube, or a `5` for a 5x5x5, etc.

But Singmaster's notation only defines moves for a 3x3x3 D: how can I turn the inner layers of a larger cube?

Unicode subscripts ($_{0123456789}$), while breaking the "a command can only be one character long" rule, give you the ability to turn the inner layers. For example, `R` (performed on a solved 4x4x4) changes the cube's state to this:

```
    0002
    0002
    0002
    0002
1111222533330444
1111222533330444
1111222533330444
1111222533330444
    5554
    5554
    5554
    5554
```

However, performing R_1 on a solved 4x4x4 changes the cube's state to this:

```
    0020
    0020
    0020
    0020
1111225233334044
1111225233334044
1111225233334044
1111225233334044
    5545
    5545
    5545
    5545
```

`R` means "turn the right face 90 ° clockwise." So R_1 means "turn the layer 1 inwards from the right face 90 ° clockwise."

These can be pretty fun. Try running $RR_1'R_2R_3'R_4\ UU_1'U_2U_3'U_4\ L'L_1L_2'L_3L_4'$ on a 5x5x5.

1.5 But how do I interact with the cube?

When I first made Cubically, with the memory cube alone and no other way to store data, it was nothing but a console emulator for a Rubik's Cube. However, I decided I wanted to turn it into an esoteric programming language, so I added one more piece of memory, the notepad. The value "written" on the notepad could be modified based on the different states of the cube. So I introduced the concept of memory locations. Memory location `0` would always store the sum of the top face of the cube (the face that's filled with `0`s when the cube is unsolved). Memory location `1` would always store the sum of the left face, `2` stored the right face sum, etc.

Try scrambling up the cube and running `:1 %6-:1` to set the notepad to the sum of the left face, and `%6` to print the notepad (memory location `6`). What did it print? Try a different scramble with the same `:1 %6` at the end. See what different values you can get!

For ease of use, Cubically automatically dumps the memory cube and the notepad at the end of the program. For example, after running the code `R2L2U2D2F2B2`, Cubically prints this:

```
Notepad: 0

   050
   505
   050
131242313424
313424131242
131242313424
   505
   050
   505
```

1.6 Cubically sounds fun, but how can I actually use it?

You can download the interpreter in the GitHub repo. You need GCC and Make installed.

You can also use the online interpreter, thanks to Dennis of Programming Puzzles and Code Golf, where you can find many fun Cubically programs in friendly competition.

1.7 Who's responsible for this mess?

Most of Cubically, such as the interpreter and the documentation, was written by me, a.k.a. MD XF on the internet (PPCG, GitHub). Thanks to the members of the Cubically team - Kamil Drakari (PPCG, GitHub) and TehPers (PPCG, GitHub) for valuable contributions.