# NestFuck

## Using the interpreter

Place a file called “program.txt” in the same directory as the interpreter. Write your NestFuck code in this file, save it, and run the interpreter. The output will be displayed through console.

If you wish to debug your program, debug information will be placed in a file called “debug.txt” in the same directory.

## Syntax

### Interpreter variables

| VARIABLE | FUNCTION |
|---|---|
| POINTER | Currently selected memory cell |
| COUNTER | Position of character being processed |
| OP | Character currently being processed |
| NESTLEVEL | Pairs of nested brackets around current character |
| OPENMARKERS | Position of open loop markers, referenced by corresponding close loop marker |
| CLOSEMARKERS | Position of close loop markers, referenced by corresponding open loop marker |
| | |

### Characters

| CHARACTER | FUNCTION |
|---|---|
| ( | Increase nestLevel by 1 |
| ) | Decrease nestLevel by 1 |
| . | Execute an instruction based on nestLevel |

### Instructions

NestFuck instructions map directly to brainfuck instructions in the following manner:

| BRAINFUCK CHARACTER | NESTLEVEL | FUNCTION |
|---|---|---|
| > | 0 | Move the pointer to the next memory cell |
| < | 1 | Move the pointer to the previous memory cell |
| + | 2 | Increment the cell under the pointer |
| - | 3 | Decrement the cell under the pointer |
| . | 4 | Output the value of the cell under the pointer as a character |
| , | 5 | Input a character and store it in the cell under the pointer |
| [ | 6 | Jump past the matching ]/7 if the cell under the pointer is 0 |
| ] | 7 | Jump back to the matching [/6 uf the cell under the pointer is not 0 |

### Errors

An error will occur if:

1) The pointer exceeds the bounds of the cell list
   a) The pointer value reduces below 0
   b) The interpreter has failed to add cells. If this happens, contact me
2) A command is executed at an invalid nesting level
   a) The nesting level exceeds 7
   b) The nesting level is less than 0
3) Brackets are unbalanced in the program
   i.e the nesting level must begin and end at 0
4) You attempt to enter multiple characters at once
   This is a non-terminal error, the interpreter will prompt you for input again

### Programming styles

Programming in NestFuck has 2 main styles.

#### Simple:

Open and close all brackets for each change in command. In simple, the brainfuck code />++++++++.</ would look like /. ((……..)) ((((.)))) (.)/. If consecutive commands are the same, the brackets are not closed between each one. Otherwise, brackets are always closed between each command.

#### Compact:

Only open and close brackets as necessary. In compact, the brainfuck code />++++++++.</ would look like /.((……..((.))).)/. The programs end up being much shorter, but much more difficult to read.

## Debugging

All debug information will be dumped into a file called debug.txt in the same directory as the interpreter. The file will contain the value of the counter, pointer, selected cell, and nest level at each character in the program. The next line contains the value of all cells. All cells that match the value of the cell under the pointer are in square brackets. This is intended to show the position of the pointer, but I couldn't be bothered to do it properly, so you will have to infer the position sometimes.

## Examples

### Hello World!

#### Full

((..... ...)) cell #0 to 8

((((((.)))))) [

    .((....)) Set cell #1 to 4

    ((((((.)))))) [

.((..)) Add 4*2 to cell #2

.((...)) Add 4*3 to cell #3

.((...)) Add 4*4 to cell #4

.((.)) Add 4 to cell #5

(....) (((.))) Decrement loop counter in cell #1

(((((((.))))))) ] Loop until cell #1 is 0

.((.)) Add 1 to cell #2

.((.)) Add 1 to cell #3

.(((.))) Subtract 1 from cell #4

..((.)) Add 1 to cell 6

((((((.)))))) [ (.) ] (((((((.))))))) Move back to cell #1

(.)(((.))) Decrement the loop counter in cell #0

(((((((.))))))) ]

..((((.)))) Output cell #2, H

.(((...)))((((.)))) Subtract 3 from cell #3 to get e

((.......))

((((..))))

((...))

((((.)))) Get llo from cell #3

..((((.)))) Space from cell #5

(.)(((.)))((((.)))) Subtract 1 from cell #4 to get W

(.)((((.)))) Get o from cell #3 again

((...))((((.))))

(((......)))

((((.))))

(((........)))((((.)))) rld from cell #3

..((.))((((.)))) ! from cell #5

.((..))((((.)))) newline

### Simple minimised

((..... ...))((((((.)))))).((....))((((((.)))))).((..)).((...)).((...)).((.))(....)
(((.)))(((((((.))))))).((.)).((.)).(((.))).((.))((((((.)))))) (.)
(((((((.)))))))(.)(((.)))(((((((.))))))).((((.)))).(((...)))((((.)))).((.......))((((..))))((...))((((.)))).((((.)))).(.)(((.)))((((.))))(.)
((((.))))((...))((((.))))(((.....)))(((.)))((((.))))(((........)))((((.)))).((.))((((.)))).((..))((((.))))

### Compact minimised

((........((((.[])))))).((....((((.[])))))).((..)).((...)).((...)).((.)....((.((((.])))))).((.)).((.)).(((.))).((.((((.[])))))).((((((.])))))).((
.((((.])))))).((((.)))).(((...(.)).......((..)).((.)))).((((.))).((.(.))).(((.)).((.)).....(.).....(.)))).((.((.)))).((.((.))))

## Cat

### Simple minimised

(((((.)))))((((((.))))))((((.))))(((((.)))))(((((((.)))))))

### Compact minimised

(((((.(.)).(.((.)))))))