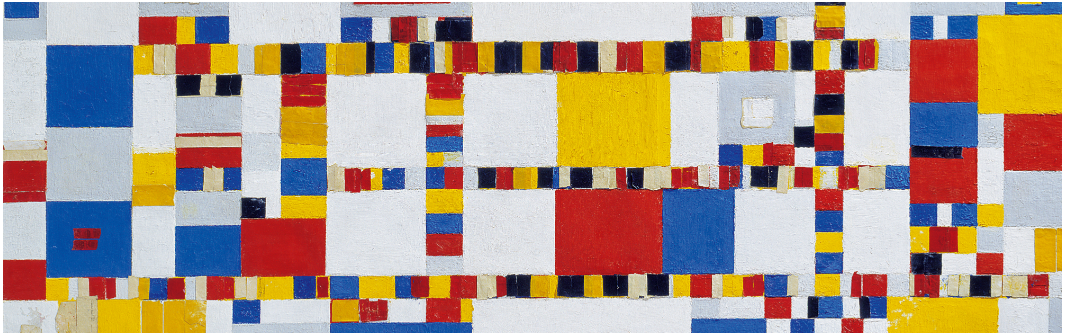


Esoteric Programming Languages

blinry

17. Esoterischprogrammiernacht

Introduction

"esoteric" from Greek *esoterikos*, "belonging to an inner circle"

Introduction

“esoteric” from Greek *esoterikos*, “belonging to an inner circle”

Esoteric Programming Languages

- proof-of-concept
- artistic expression
- challenge (for the designer and/or user)
- joke

Introduction

“esoteric” from Greek *esoterikos*, “belonging to an inner circle”

Esoteric Programming Languages

- proof-of-concept
- artistic expression
- challenge (for the designer and/or user)
- joke

This presentation: Five Turing-complete languages

Brainfuck

- Designed by Urban Müller in 1993

Brainfuck

- Designed by Urban Müller in 1993
- Motivation: Small compiler (296 bytes)

Brainfuck

- Designed by Urban Müller in 1993
- Motivation: Small compiler (296 bytes)
- **Minimalist syntax**, only eight commands:
  `> < + - , . [ ]`

Brainfuck

- Designed by Urban Müller in 1993
- Motivation: Small compiler (296 bytes)
- **Minimalist syntax**, only eight commands:
  `> < + - , . [ ]`
- “brain fuck” = hard or complicated thing

Examples

```
>>>,-----.<<<++++++++++.
```

Examples

```
>>>,-----.<<<++++++++++.
```

Examples

```
>>>,-----.<<<++++++++++.
```

Examples

```
>>>,-----.<<<++++++++++.
```

Examples

```
>>>,-----.<<<++++++++++.
```

Examples

```
>>>,-----.<<<++++++++++.
```

Examples

```
>>>,-----.<<<++++++++++.
```

Examples

```
>>>,-----.<<<++++++++++.
```

Examples

```
>>>,-----.<<<++++++++++.
```

Input: `f`

Examples

```
>>>,-----.<<<++++++++++.
```

Input: `f` Output: `a`$\hookleftarrow$

Examples

```
>>>,-----.<<<++++++++++.
```

Input: `f` Output: `a`$\hookleftarrow$

```
+++++[->+<]
```

Examples

```
>>>,-----.<<<++++++++++.
```

Input: `f` Output: `a`$\hookleftarrow$

```
+++++[->+<]
```

Examples

```
>>>,-----.<<<+++++++++++.
```

Input: `f` Output: `a`$\hookleftarrow$

```
+++++[->+<]
```

Examples

```
>>>,-----.<<<++++++++++.
```

Input: `f` Output: `a`$\hookleftarrow$

```
+++++[->+<]
```

Examples

```
>>>,-----.<<<++++++++++.
```

Input: `f` Output: `a`$\hookleftarrow$

```
+++++[->+<]
```

Examples

```
>>>,-----.<<<++++++++++.
```

Input: `f` Output: `a`$\hookleftarrow$

```
+++++[->+<]
```

Examples

```
>>>,-----.<<<++++++++++.
```

Input: `f` Output: `a`$\hookleftarrow$

```
+++++[->+<]
```

Examples

```
>>>,-----.<<<++++++++++.
```

Input: `f` Output: `a`$\hookleftarrow$

```
+++++[->+<]
```

Examples

```
>>>,-----.<<<++++++++++.
```

Input: `f` Output: `a`$\hookleftarrow$

```
+++++[->+<]
```

```
-[>,[<<[-<]++[->+]->-]<<<<
<<<<<+[-<+++++++++[->++++++
<]>.[-]>+]-]
```

Examples

```
>>>,-----.<<<++++++++++.
```

Input: `f` Output: `a`$\hookleftarrow$

```
+++++[->+<]
```

```
-[>,[<<[-<]++[->+]->-]<<<<
<<<<<+[-<+++++++++[->++++++
<]>.[-]>+]-]
```

Input: `h`

Examples

```
>>>,-----.<<<++++++++++.
```

Input: `f` Output: `a`↩

```
+++++[->+<]
```

```
-[>,[<<[-<]++[->+]->-]<<<<
<<<<<+[-<++++++++[->++++++
<]>.[-]>+]-]
```

Input: `h` Output: `01101000`

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

... 0 0 0 0 0 0 0 0 **0** 0 ...

Input

`hello`

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

... 0 0 0 0 0 0 0 0 **255** 0 ...

Input

```
hello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... 0 0 0 0 0 0 0 0 **255** 0 ...

Input

```
hello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

```
...   0   0   0   0   0   0   0   0   255   0   ...
```

Input

```
hello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

```
...  0  0  0  0  0  0  0  0  255  104  ...
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

```
...  0  0  0  0  0  0  0  0  255  104  ...
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

... 0 0 0 0 0 0 0 0 **255** 104 ...

Input

`ello`

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... 0 0 0 0 0 0 0 **0** 255 104 ...

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... 0 0 0 0 0 0 0 **0** 255 104 ...

Input

`ello`

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... 0 0 0 0 0 0 0 **0** 255 104 ...

Input

ello

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

```
...  0  0  0  0  0  0  0  1  255 104 ...
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... 0 0 0 0 0 0 0 2 255 104 ...

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

```
... 0 0 0 0 0 0 0 2 255 104 ...
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... 0 0 0 0 0 0 0 1 255 104 ...

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

```
...  0  0  0  0  0  0  0  1  255 104 ...
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... 0 0 0 0 0 0 0 1 **0** 104 ...

Input

ello

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

. . . 0 0 0 0 0 0 0 1 **0** 104 . . .

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

```
...  0  0  0  0  0  0  0  1  255  104  ...
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

... 0 0 0 0 0 0 0 1 255 **104** ...

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

... 0 0 0 0 0 0 0 1 255 **103** ...

Input

ello

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

... 0 0 0 0 0 0 0 1 255 **103** ...

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

```
. . .  0  0  0  0  0  0  0  1  255  103  . . .
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

. . . 0 0 0 0 0 0 0 1 **255** 103 . . .

Input

`ello`

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

... 0 0 0 0 0 0 0 **1** 255 103 ...

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

```
...  0  0  0  0  0  0  0  1  255 103 ...
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... 0 0 0 0 0 0 0 **0** 255 103 ...

Input

ello

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

```
...  0  0  0  0  0  0  0  0  255 103 ...
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... 0 0 0 0 0 0 **0** 0 255 103 ...

Input

`ello`

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

```
...  0  0  0  0  0  0  1  0  255 103 ...
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

. . . 0 0 0 0 0 0 **2** 0 255 103 . . .

Input

`ello`

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

. . . 0 0 0 0 0 0 **2** 0 255 103 . . .

Input

`ello`

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

```
...  0   0   0   0   0   0   1   0  255 103 ...
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

```
...  0  0  0  0  0  0  1  0  255 103 ...
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

... 0 0 0 0 0 0 1 1 255 103 ...

Input

ello

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

```
...  0  0  0  0  0  0  1  1  255 103 ...
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... 0 0 0 0 0 0 1 **1** 255 103 ...

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... 0 0 0 0 0 0 1 **0** 255 103 ...

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

... 0 0 0 0 0 0 1 0 **255** 103 ...

Input

ello

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... 0 0 0 0 0 0 1 0 **0** 103 ...

Input

ello

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... 0 0 0 0 0 0 1 0 **0** 103 ...

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

```
...   0   0   0   0   0   0   1   0  255 103 ...
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

. . .   0   0   0   0   0   0   1   0   255   **103**   . . .

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... 0 0 0 0 0 0 1 0 255 **102** ...

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
  >,
  [
    <<
    [-<]
    +
    +[->+]-
    >-
  ]
  <<<<<<<<<
  +[-
    <++++++++
    [->++++++<]>.
    [-]
    >
  +]-
]
```

Tape

```
...   0   0   0   0   0   0   1   0  255 102 ...
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

```
...  0  1  1  0  1  0  0  0  255  0  ...
```

Input

```
ello
```

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... **0** 1 1 0 1 0 0 0 255 0 ...

Input

`ello`

Output

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

```
...  0  0  0  0  0  0  0  0  255  0  ...
```

Input

```
ello
```

Output

```
01101000
```

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

. . . 0 0 0 0 0 0 0 0 **255** 0 . . .

Input

```
ello
```

Output

```
01101000
```

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

... 0 0 0 0 0 0 0 0 **255** 0 ...

Input

```
ello
```

Output

```
01101000
```

Example: ASCII to binary

```
-
[
   >,
   [
      <<
      [-<]
      +
      +[->+]-
      >-
   ]
   <<<<<<<<<
   +[-
      <++++++++
      [->++++++<]>.
      [-]
      >
   +]-
]
```

Tape

```
. . .   0   0   0   0   0   0   0   0   255   0   . . .
```

Input

Output

```
0110100001100101011011000110110001101111
```

Significance

- Best-known esoteric programming language

Significance

- Best-known esoteric programming language
- Many implementations, like *Awib*

Significance

- Best-known esoteric programming language
- Many implementations, like *Awib*
- Smallest current interpreter: 98 bytes!

Significance

- Best-known esoteric programming language
- Many implementations, like *Awib*
- Smallest current interpreter: 98 bytes!
- Someone wrote a text adventure:

Significance

- Best-known esoteric programming language
- Many implementations, like *Awib*
- Smallest current interpreter: 98 bytes!
- Someone wrote a text adventure:

```
>+>+>+>+>+>+>>>>++++++++++[>+>+++>+++++>+++++++>++++++++++<<
<<<-]>..>++>>.>+++++.+++++.-------------.+++++++++++..++++++
+++++++.<<<.>>+++++++++++++.>-------------------------.+++++
+++++.--.+++++.-------.<<<.>>-----.>+++++++++++++++.--------
---------.+++++++++++++.<<<.>>+++++.>----------.---.<<<.>>++
+.>++++++++++.+++.------.--------.<<<<.>>-----..............
...............<<..>>>-------.++++++++.-----------.+++++++++
++++.---------.+++++++++++++++.<<+++++++++++++.<.>>-.>--.++.
---------------.<<<<.>>>------------.>++++++++++.-.++++++.--
---------.+++++.---------.+++++++++.++++++.<<.<.>>++++++++++
```

Variants

At least 200 variants, including:

Variants

At least 200 variants, including:

- **DoubleFuck** (two tapes)

Variants

At least 200 variants, including:

- **DoubleFuck** (two tapes)
- **Boolfuck** (binary cells)

Variants

At least 200 variants, including:

- **DoubleFuck** (two tapes)
- **Boolfuck** (binary cells)
- **Brainfork** (multithreading via `Y`)

Variants

At least 200 variants, including:

- **DoubleFuck** (two tapes)
- **Boolfuck** (binary cells)
- **Brainfork** (multithreading via `Y`)
- **Ook!** (for orangutans)

INTERCAL

- Created in 1972 by Donald R. Woods and James M. Lyon

INTERCAL

- Created in 1972 by Donald R. Woods and James M. Lyon
- Motivation: Be different than FORTRAN or COBOL

INTERCAL

- Created in 1972 by Donald R. Woods and James M. Lyon
- Motivation: Be different than FORTRAN or COBOL
- **Weird** names, operators and properties

INTERCAL

- Created in 1972 by Donald R. Woods and James M. Lyon
- Motivation: Be different than FORTRAN or COBOL
- **Weird** names, operators and properties
- “Compiler Language With No Pronounceable Acronym”

INTERCAL

- Created in 1972 by Donald R. Woods and James M. Lyon
- Motivation: Be different than FORTRAN or COBOL
- **Weird** names, operators and properties
- “Compiler Language With No Pronounceable Acronym”
- Fun manual!

Manual

- “Under no circumstances confuse the mesh with the interleave operator, except under confusing circumstances!”

Manual

- “Under no circumstances confuse the mesh with the interleave operator, except under confusing circumstances!”
- “Definition of array dimensions will be discussed later in greater detail, since discussing it in less detail would be difficult.”

Manual

- "Under no circumstances confuse the mesh with the interleave operator, except under confusing circumstances!"
- "Definition of array dimensions will be discussed later in greater detail, since discussing it in less detail would be difficult."
- "*exp* represents any expression (except colloquial and facial expressions)"

Manual

- “Under no circumstances confuse the mesh with the interleave operator, except under confusing circumstances!”
- “Definition of array dimensions will be discussed later in greater detail, since discussing it in less detail would be difficult.”
- “*exp* represents any expression (except colloquial and facial expressions)”
- “Precedence of operators is as follows: (The remainder of this page intentionally left blank.)”

Manual

- “Under no circumstances confuse the mesh with the interleave operator, except under confusing circumstances!”
- “Definition of array dimensions will be discussed later in greater detail, since discussing it in less detail would be difficult.”
- “*exp* represents any expression (except colloquial and facial expressions)”
- “Precedence of operators is as follows: (The remainder of this page intentionally left blank.)”
- “This footnote intentionally unreferenced.”

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

```
FIVE FOUR
```

```
.1 = 110110
```

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

FIVE FOUR

```
.1 = 110110
```

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

```
FIVE FOUR
```

```
.1 = 110110
```

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

```
FIVE FOUR
```

```
.1 = 110110
:1 = 110110 $ 000000
```

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

FIVE FOUR

```
.1 = 110110
:1 = 101000101000
```

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

```
FIVE FOUR
```

```
.1 = 110110
:1 = 101000101000
```

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

```
FIVE FOUR
```

```
.1 = 110110
:1 = 101000101000
:2 = ...111111 $ ...000001
```

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

```
FIVE FOUR
```

```
.1 = 110110
:1 = 101000101000
:2 = 101010101011
```

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

```
FIVE FOUR
```

```
.1 = 110110
:1 = 101000101000
:2 = 101010101011
```

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

```
FIVE FOUR
```

```
.1 = 1 1 0 1 1 00
:1 = 101000101000
:2 = 101010101011
```

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

FIVE FOUR

```
.1 = 1 1 0 1 1 00
:1 = 101000101000
:2 = 101010101011
```

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

```
FIVE FOUR
```

```
.1 = 1 1 0 1 1 00
:1 = 101000101000
:2 = 101010101011
```

Output

```
CVIII
```

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

```
FIVE FOUR
```

```
.1 = 1 1 0 1 1 00
:1 = 101000101000
:2 = 101010101011
```

Output

```
CVIII
CCXVI
CDXXXII
...
ICL275I: DON'T BYTE OFF MORE THAN
YOU CAN CHEW
```

Example: Multiplying by two

```
PLEASE WRITE IN .1

DO COME FROM (42)

DO :1 <- .1$#0
DO :2 <- #65535$#1
DO .1 <- :1~:2

(42) DO READ OUT .1

PLEASE GIVE UP
```

Input

```
FIVE FOUR
```

```
.1 = 1 1 0 1 1 00
:1 = 101000101000
:2 = 101010101011
```

Output

```
CVIII
CCXVI
CDXXXII
...
ICL275I: DON'T BYTE OFF MORE THAN
YOU CAN CHEW
```

Significance

- Eric Raymond released **C-INTERCAL** in 1990

Significance

- Eric Raymond released **C-INTERCAL** in 1990
- “Large”, active community

Significance

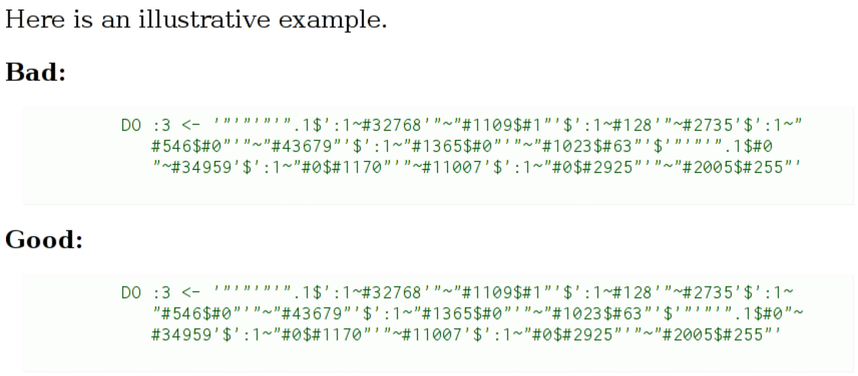
- Eric Raymond released **C-INTERCAL** in 1990
- “Large”, active community
- Donald Knuth wrote a bug report in 2010

Significance

- Eric Raymond released **C-INTERCAL** in 1990
- “Large”, active community
- Donald Knuth wrote a bug report in 2010
- Google released a style guide!

Significance

- Eric Raymond released **C-INTERCAL** in 1990
- “Large”, active community
- Donald Knuth wrote a bug report in 2010
- Google released a style guide!



Here is an illustrative example.

Bad:

```
DO :3 <- '"'"'"'".1$':1~#32768'"~"#1109$#1"'$':1~#128'"~#2735'$':1~"
   #546$#0"'"~"#43679"'$':1~"#1365$#0"'"~"#1023$#63"'$'"'"'".1$#0
   "~#34959'$':1~"#0$#1170"'"~#11007'$':1~"#0$#2925"'"~"#2005$#255"'
```

Good:

```
DO :3 <- '"'"'"'".1$':1~#32768'"~"#1109$#1"'$':1~#128'"~#2735'$':1~
   "#546$#0"'"~"#43679"'$':1~"#1365$#0"'"~"#1023$#63"'$'"'"'".1$#0"~
   #34959'$':1~"#0$#1170"'"~#11007'$':1~"#0$#2925"'"~"#2005$#255"'
```

Variants

- **TriINTERCAL** (operates on ternary values)

Variants

- **TriINTERCAL** (operates on ternary values)
- **Threaded INTERCAL** (handles `COME FROM` statements referencing the same line)

Variants

- **TriINTERCAL** (operates on ternary values)
- **Threaded INTERCAL** (handles `COME FROM` statements referencing the same line)
- **Backtracking INTERCAL** (introduces the `MAYBE` label)

Befunge

- Created in 1993 by Chris Pressey

Befunge

- Created in 1993 by Chris Pressey
- Motivation: be difficult to parse

Befunge

- Created in 1993 by Chris Pressey
- Motivation: be difficult to parse
- First **two-dimensional** language

Befunge

- Created in 1993 by Chris Pressey
- Motivation: be difficult to parse
- First **two-dimensional** language
- “Befunge” mistyping of “before”

Examples

```
>v
^<
```

Examples

```
>v
^<
```

Examples

```
>v
^<
```

Examples

```
>v
^<
```

Examples

```
>v
^<
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Output: 1011110010...

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Output: 1011110010...

```
666*+.@
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Output: 1011110010...

```
666*+.@
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Output: 1011110010. . .

```
666*+.@
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Output: 1011110010. . .

```
666*+.@
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Output: 1011110010...

```
666*+.@
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Output: 1011110010. . .

```
666*+.@
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Output: 1011110010. . .

```
666*+.@
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Output: 1011110010...

```
666*+.@
```

Examples

```
>v
^<
```

```
v>0 v
>?<.<
 >1 ^
```

Output: 1011110010. . .

```
666*+.@
```

Output: 42

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

(empty)

Input

`hello, world!`

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

(empty)

Input

`hello, world!`

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^    -4-9<
         |          <
^    +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^    -4-9<
         |          <
^    +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 **104**

Input

```
 ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 104

Input

```
 ello, world!
```

Output

Example: ROT13

```
>~:"‘"‘!v
,v      _:"z"‘v
^             _:"m"‘v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"‘^
^       _:"Z"‘^
 >:"@"‘!^
```

Stack (bottom → top)

104 104 **96**

Input

```
 ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 104 96

Input

```
 ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 1

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 **0**

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 0

Input

```
 ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^    -4-9<
        |           <
^    +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 **104**

Input

```
 ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^    -4-9<
        |           <
^    +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 104

Input

```
 ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 104 **122**

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 104 122

Input

```
 ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 **0**

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"‘"‘!v
,v      _:"z"‘v
^             _:"m"‘v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"‘^
^       _:"Z"‘^
 >:"@"‘!^
```

Stack (bottom → top)

104 0

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"‘"‘!v
,v      _:"z"‘v
^             _:"m"‘v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"‘^
^       _:"Z"‘^
 >:"@"‘!^
```

Stack (bottom → top)

104

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 **104**

Input

`ello, world!`

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 104

Input

```
 ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 104 **109**

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 104 109

Input

```
 ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 **0**

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"‘"‘!v
,v      _:"z"‘v
^             _:"m"‘v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"‘^
^       _:"Z"‘^
 >:"@"‘!^
```

Stack (bottom → top)

104 0

Input

```
 ello, world!
```

Output

Example: ROT13

```
>~:"‘"‘!v
,v      _:"z"‘v
^             _:"m"‘v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"‘^
^       _:"Z"‘^
 >:"@"‘!^
```

Stack (bottom → top)

104 0

Input

```
 ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^    -4-9<
        |           <
^    +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

104 **9**

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

113

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

113 **4**

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

117

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

117

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:":Z"`^
 >:"@"`!^
```

Stack (bottom → top)

117

Input

```
 ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

117

Input

```
ello, world!
```

Output

Example: ROT13

```
>~:"`"`!v
,v       _:"z"`v
^              _:"m"`v
^    -4-9<
         |           <
^    +4+9<
^              _:"M"`^
^        _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

(empty)

Input

```
ello, world!
```

Output

```
u
```

Example: ROT13

```
>~:"`"`!v
,v      _:"z"`v
^             _:"m"`v
^   -4-9<
        |           <
^   +4+9<
^             _:"M"`^
^       _:"Z"`^
 >:"@"`!^
```

Stack (bottom → top)

(empty)

Input

Output

```
uryyb, jbeyq!
```

Significance

- Important platform: *Befunge Mailing List*

Significance

- Important platform: *Befunge Mailing List*
- Many actively maintained interpreters and compilers, like *befunjit*

Significance

- Important platform: *Befunge Mailing List*
- Many actively maintained interpreters and compilers, like *befunjit*
- IRC client with 10,000 characters

Variants

- Other members of the *Funge-98* family: **Unefunge** and **Trefunge**

Variants

- Other members of the *Funge-98* family: **Unefunge** and **Trefunge**
- **Weird** (only one instruction)

Variants

- Other members of the *Funge-98* family: **Unefunge** and **Trefunge**
- **Weird** (only one instruction)
- **PATH** (1D source code, 2D playfield)

Malbolge

- Created in 1998 by Ben Olmstead

Malbolge

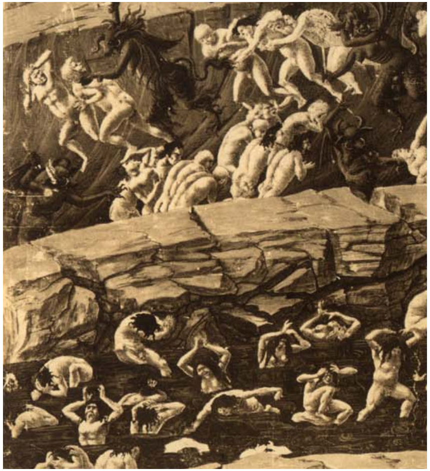
- Created in 1998 by Ben Olmstead
- Motivation: be incomprehensible and **hard** to use

Malbolge

- Created in 1998 by Ben Olmstead
- Motivation: be incomprehensible and **hard** to use
- Took two years to write the first nontrivial program

Malbolge

- Created in 1998 by Ben Olmstead
- Motivation: be incomprehensible and **hard** to use
- Took two years to write the first nontrivial program
- *Malebolge* is the eighth circle of Hell in Dante's *Inferno*



Description

- Simple virtual machine

Description

- Simple virtual machine
- CPU with three registers A, C, and D

Description

- Simple virtual machine
- CPU with three registers A, C, and D
- 3^{10} memory cells, 10 trits each

Description

- Simple virtual machine
- CPU with three registers A, C, and D
- 3^{10} memory cells, 10 trits each

Execution

For each instruction:

Description

- Simple virtual machine
- CPU with three registers A, C, and D
- 3^{10} memory cells, 10 trits each

Execution

For each instruction:

- Subtract 33, add C, mod with 94

Description

- Simple virtual machine
- CPU with three registers A, C, and D
- 3^{10} memory cells, 10 trits each

Execution

For each instruction:

- Subtract 33, add C, mod with 94
- Apply a substitution encryption

Description

- Simple virtual machine
- CPU with three registers A, C, and D
- 3^{10} memory cells, 10 trits each

Execution

For each instruction:

- Subtract 33, add C, mod with 94
- Apply a substitution encryption
- If we now have one of `j i * p / < v o`, execute that instruction

Description

- Simple virtual machine
- CPU with three registers A, C, and D
- 3^{10} memory cells, 10 trits each

Execution

For each instruction:

- Subtract 33, add C, mod with 94
- Apply a substitution encryption
- If we now have one of `j i * p / < v o`, execute that instruction
- Subtract 33

Description

- Simple virtual machine
- CPU with three registers A, C, and D
- 3^{10} memory cells, 10 trits each

Execution

For each instruction:

- Subtract 33, add C, mod with 94
- Apply a substitution encryption
- If we now have one of `j i * p / < v o`, execute that instruction
- Subtract 33
- Apply a different substitution encryption

Description

- Simple virtual machine
- CPU with three registers A, C, and D
- 3^{10} memory cells, 10 trits each

Execution

For each instruction:

- Subtract 33, add C, mod with 94
- Apply a substitution encryption
- If we now have one of `j i * p / < v o`, execute that instruction
- Subtract 33
- Apply a different substitution encryption
- Increment C and D

Example: Hello world

```
(=<`$9]7<5YXz7wT.3,+O/o'K%$H"'~D|#z@b=`{^Lx8%$X
mrkpohm-kNi;gsedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFE
DCBA@?>=<;:9876543s+O<oLm
```

Example: Hello world

```
(=<`$9]7<5YXz7wT.3,+O/o'K%$H"'~D|#z@b=`{^Lx8%$X
mrkpohm-kNi;gsedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFE
DCBA@?>=<;:9876543s+O<oLm
```

Output

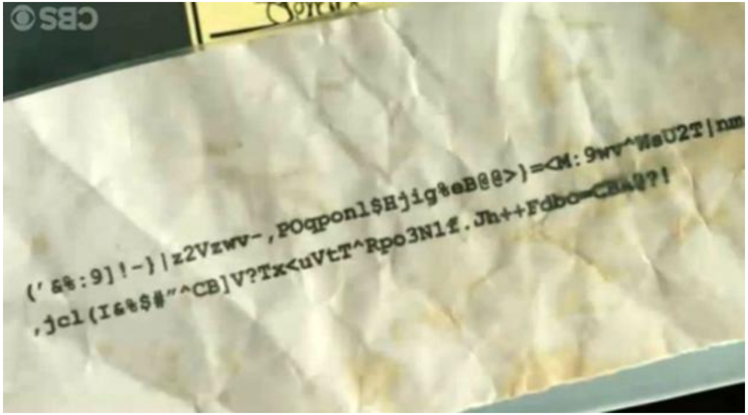
```
HEllO WORld
```

Significance

- Cryptanalysis by Louis Scheffer around 2005

Significance

- Cryptanalysis by Louis Scheffer around 2005
- Appearance in *Elementary* S01E10:



```
('&%:9]!~}|z2Vzwv-,POqponl$Hjig%eB@@>}=<M:9wv^WsU2T|nm
,jcl(I&%$#"^CB]V?Tx<uVtT^Rpo3Nlf.Jh++Fdbo=CBA@?!
```

Shakespeare

- Created in 2001 by Karl Hasselström and Jon Åslund

Shakespeare

- Created in 2001 by Karl Hasselström and Jon Åslund
- Motivation: homework in their Syntax Analysis class

Shakespeare

- Created in 2001 by Karl Hasselström and Jon Åslund
- Motivation: homework in their Syntax Analysis class
- **Themed** language

Shakespeare

- Created in 2001 by Karl Hasselström and Jon Åslund
- Motivation: homework in their Syntax Analysis class
- **Themed** language
- “combines the expressiveness of BASIC with the user-friendliness of assembly language”

Example: Fibonacci sequence

```
A drama by the numbers.

Juliet, a young Italian lady.
Romeo, the rich Count.
Mercutio, his spacy rival.
```

Variables

Output

Example: Fibonacci sequence

```
A drama by the numbers.

Juliet, a young Italian lady.
Romeo, the rich Count.
Mercutio, his spacy rival.
```

Variables

Output

Example: Fibonacci sequence

```
A drama by the numbers.

Juliet, a young Italian lady.
Romeo, the rich Count.
Mercutio, his spacy rival.
```

Variables

Juliet	0	off

Output

Example: Fibonacci sequence

```
A drama by the numbers.

Juliet, a young Italian lady.
Romeo, the rich Count.
Mercutio, his spacy rival.
```

Variables

Juliet	0	off
Romeo	0	off

Output

Example: Fibonacci sequence

```
A drama by the numbers.

Juliet, a young Italian lady.
Romeo, the rich Count.
Mercutio, his spacy rival.
```

Variables

Juliet	0	off
Romeo	0	off
Mercutio	0	off

Output

Example: Fibonacci sequence

```
    Act I: The Act where it all happens.
     Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as
  beautiful as a flower!
Juliet: You are a disgusting smelly lying
  rotten dirty pig! You are as small as
  the difference between nothing and
  thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
```

Variables

Juliet	0	off
Romeo	0	off
Mercutio	0	off

Output

Example: Fibonacci sequence

```
   Act I: The Act where it all happens.
    Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as
  beautiful as a flower!
Juliet: You are a disgusting smelly lying
  rotten dirty pig! You are as small as
  the difference between nothing and
  thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
```

Variables

Juliet	0	off
Romeo	0	off
Mercutio	0	off

Output

Example: Fibonacci sequence

```
    Act I: The Act where it all happens.
     Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as
  beautiful as a flower!
Juliet: You are a disgusting smelly lying
  rotten dirty pig! You are as small as
  the difference between nothing and
  thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
```

Variables

Juliet	0	on
Romeo	0	off
Mercutio	0	on

Output

Example: Fibonacci sequence

```
    Act I: The Act where it all happens.
     Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as
  beautiful as a flower!
Juliet: You are a disgusting smelly lying
  rotten dirty pig! You are as small as
  the difference between nothing and
  thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
```

Variables

Juliet	2	on
Romeo	0	off
Mercutio	0	on

Output

Example: Fibonacci sequence

```
    Act I: The Act where it all happens.
     Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as
  beautiful as a flower!
Juliet: You are a disgusting smelly lying
  rotten dirty pig! You are as small as
  the difference between nothing and
  thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
```

Variables

Juliet	1	on
Romeo	0	off
Mercutio	0	on

Output

Example: Fibonacci sequence

```
    Act I: The Act where it all happens.
     Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as
  beautiful as a flower!
Juliet: You are a disgusting smelly lying
  rotten dirty pig! You are as small as
  the difference between nothing and
  thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
```

Variables

Juliet	1	on
Romeo	0	off
Mercutio	-32	on

Output

Example: Fibonacci sequence

```
    Act I: The Act where it all happens.
     Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as
  beautiful as a flower!
Juliet: You are a disgusting smelly lying
  rotten dirty pig! You are as small as
  the difference between nothing and
  thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
```

Variables

Juliet	1	on
Romeo	0	off
Mercutio	32	on

Output

Example: Fibonacci sequence

```
    Act I: The Act where it all happens.
     Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as
  beautiful as a flower!
Juliet: You are a disgusting smelly lying
  rotten dirty pig! You are as small as
  the difference between nothing and
  thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
```

Variables

Juliet	1	on
Romeo	0	off
Mercutio	32	off

Output

Example: Fibonacci sequence

```
   Act I: The Act where it all happens.
    Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as
  beautiful as a flower!
Juliet: You are a disgusting smelly lying
  rotten dirty pig! You are as small as
  the difference between nothing and
  thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
```

Variables

Juliet	1	on
Romeo	0	on
Mercutio	32	off

Output

Example: Fibonacci sequence

```
    Act I: The Act where it all happens.
     Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as
  beautiful as a flower!
Juliet: You are a disgusting smelly lying
  rotten dirty pig! You are as small as
  the difference between nothing and
  thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
```

Variables

Juliet	1	on
Romeo	-1	on
Mercutio	32	off

Output

Example: Fibonacci sequence

```
   Act I: The Act where it all happens.
    Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as
  beautiful as a flower!
Juliet: You are a disgusting smelly lying
  rotten dirty pig! You are as small as
  the difference between nothing and
  thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
```

Variables

Juliet	1	on
Romeo	**0**	on
Mercutio	32	off

Output

Example: Fibonacci sequence

```
    Act I: The Act where it all happens.
     Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as
  beautiful as a flower!
Juliet: You are a disgusting smelly lying
  rotten dirty pig! You are as small as
  the difference between nothing and
  thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
```

Variables

Juliet	1	on
Romeo	0	on
Mercutio	32	off

Output

```
1
```

Example: Fibonacci sequence

```
    Act I: The Act where it all happens.
     Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as
  beautiful as a flower!
Juliet: You are a disgusting smelly lying
  rotten dirty pig! You are as small as
  the difference between nothing and
  thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
```

Variables

Juliet	1 **(0)**	on
Romeo	0	on
Mercutio	32	off

Output

```
1
```

Example: Fibonacci sequence

```
    Act I: The Act where it all happens.
     Scene I: Juliet insults everyone.

[Enter Juliet and Mercutio]

Mercutio: You charming angel! You are as
  beautiful as a flower!
Juliet: You are a disgusting smelly lying
  rotten dirty pig! You are as small as
  the difference between nothing and
  thyself!

[Exit Mercutio]
[Enter Romeo]

Juliet: You devil! You are nothing!
Romeo: Open your heart! Remember me!

[Exit Juliet]
```

Variables

Juliet	1 (0)	off
Romeo	0	on
Mercutio	32	off

Output

```
1
```

Example: Fibonacci sequence

```
      Scene II: The rival's encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let
  us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum
  of thyself and a stone wall! Remember
  yourself!

[Exit Mercutio]
```

Variables

Juliet	1 (0)	off
Romeo	0	on
Mercutio	32	off

Output

```
1
```

Example: Fibonacci sequence

```
      Scene II: The rival's encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let
  us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum
  of thyself and a stone wall! Remember
  yourself!

[Exit Mercutio]
```

Variables

Juliet	1 (0)	off
Romeo	0	on
Mercutio	32	on

Output

```
1
```

Example: Fibonacci sequence

```
      Scene II: The rival's encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let
  us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum
  of thyself and a stone wall! Remember
  yourself!

[Exit Mercutio]
```

Variables

Juliet	1 (0)	off
Romeo	0	on
Mercutio	32	on

Output

```
1
```

Example: Fibonacci sequence

```
      Scene II: The rival's encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let
  us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum
  of thyself and a stone wall! Remember
  yourself!

[Exit Mercutio]
```

Variables

Juliet	1 (0)	off
Romeo	0	on
Mercutio	32	on

Output

```
1
```

Example: Fibonacci sequence

```
      Scene II: The rival's encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let
  us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum
  of thyself and a stone wall! Remember
  yourself!

[Exit Mercutio]
```

Variables

Juliet	1 (0)	off
Romeo	0	on
Mercutio	32	on

Output

```
1␣
```

Example: Fibonacci sequence

```
      Scene II: The rival's encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let
  us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum
  of thyself and a stone wall! Remember
  yourself!

[Exit Mercutio]
```

Variables

Juliet	1 (0)	off
Romeo	**1**	on
Mercutio	32	on

Output

```
1␣
```

Example: Fibonacci sequence

```
      Scene II: The rival's encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let
  us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum
  of thyself and a stone wall! Remember
  yourself!

[Exit Mercutio]
```

Variables

Juliet	1 (0)	off
Romeo	1 **(1)**	on
Mercutio	32	on

Output

```
1␣
```

Example: Fibonacci sequence

```
      Scene II: The rival's encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let
  us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum
  of thyself and a stone wall! Remember
  yourself!

[Exit Mercutio]
```

Variables

Juliet	1 (0)	off
Romeo	1 (1)	on
Mercutio	32	off

Output

```
1␣
```

Example: Fibonacci sequence

```
    Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as
  happy as the sum of thyself and me. Open
  your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!
```

Variables

Juliet	1 (0)	off
Romeo	1 (1)	on
Mercutio	32	off

Output

```
1␣
```

Example: Fibonacci sequence

```
    Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as
  happy as the sum of thyself and me. Open
  your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!
```

Variables

Juliet	1 (0)	on
Romeo	1 (1)	on
Mercutio	32	off

Output

```
1␣
```

Example: Fibonacci sequence

```
    Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as
  happy as the sum of thyself and me. Open
  your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!
```

Variables

Juliet	1 (0)	on
Romeo	1 (1)	on
Mercutio	32	off

Output

```
1␣
```

Example: Fibonacci sequence

```
    Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as
  happy as the sum of thyself and me. Open
  your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!
```

Variables

Juliet	**0**	on
Romeo	1 (1)	on
Mercutio	32	off

Output

```
1␣
```

Example: Fibonacci sequence

```
    Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as
  happy as the sum of thyself and me. Open
  your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!
```

Variables

Juliet	**1**	on
Romeo	1 (1)	on
Mercutio	32	off

Output

```
1␣
```

Example: Fibonacci sequence

```
    Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as
  happy as the sum of thyself and me. Open
  your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!
```

Variables

Juliet	1	on
Romeo	1 (1)	on
Mercutio	32	off

Output

```
1␣1
```

Example: Fibonacci sequence

```
    Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as
  happy as the sum of thyself and me. Open
  your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!
```

Variables

Juliet	1 **(1)**	on
Romeo	1 (1)	on
Mercutio	32	off

Output

```
1␣1
```

Example: Fibonacci sequence

```
    Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as
  happy as the sum of thyself and me. Open
  your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!
```

Variables

Juliet	1 (1)	on
Romeo	1	on
Mercutio	32	off

Output

```
1␣1
```

Example: Fibonacci sequence

```
    Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as
  happy as the sum of thyself and me. Open
  your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!
```

Variables

Juliet	1 (1)	off
Romeo	1	on
Mercutio	32	off

Output

```
1␣1
```

Example: Fibonacci sequence

```
    Scene III: Can I have your number?

[Enter Juliet]

Juliet: You are me!

Romeo: Recall our eternal love! You are as
  happy as the sum of thyself and me. Open
  your heart! Remember me!

Juliet: Recall that we all must die.

[Exit Juliet]

Romeo: We must return to scene II!
```

Variables

Juliet	1 (1)	off
Romeo	1	on
Mercutio	32	off

Output

```
1␣1
```

Example: Fibonacci sequence

```
      Scene II: The rival's encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let
  us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum
  of thyself and a stone wall! Remember
  yourself!

[Exit Mercutio]
```

Variables

Juliet	1 (1)	off
Romeo	1	on
Mercutio	32	off

Output

```
1␣1
```

Example: Fibonacci sequence

```
        Scene II: The rival's encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let
  us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum
  of thyself and a stone wall! Remember
  yourself!

[Exit Mercutio]
```

Variables

Juliet	**3524578**	off
Romeo	**32**	on
Mercutio	32	off

Output

```
1 1 2 3 5 8 13 21 34 55 89 144
233 377 610 987 1597 2584 4181
6765 10946 17711 28657 46368
75025 121393 196418 317811 514229
832040 1346269 2178309 3524578
```

Example: Fibonacci sequence

```
      Scene II: The rival's encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let
  us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum
  of thyself and a stone wall! Remember
  yourself!

[Exit Mercutio]
```

Variables

Juliet	3524578	off
Romeo	32	on
Mercutio	32	on

Output

```
1 1 2 3 5 8 13 21 34 55 89 144
233 377 610 987 1597 2584 4181
6765 10946 17711 28657 46368
75025 121393 196418 317811
514229 832040 1346269 2178309
3524578
```

Example: Fibonacci sequence

```
      Scene II: The rival's encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let
  us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum
  of thyself and a stone wall! Remember
  yourself!

[Exit Mercutio]
```

Variables

Juliet	3524578	off
Romeo	32	on
Mercutio	32	on

Output

```
1 1 2 3 5 8 13 21 34 55 89 144

233 377 610 987 1597 2584 4181

6765 10946 17711 28657 46368

75025 121393 196418 317811

514229 832040 1346269 2178309

3524578
```

Example: Fibonacci sequence

```
      Scene II: The rival's encounter.

[Enter Mercutio]

Romeo: Are you better than me? If not, let
  us proceed to scene IV. Speak your mind!

Mercutio: You are as miserable as the sum
  of thyself and a stone wall! Remember
  yourself!

[Exit Mercutio]
```

Variables

Juliet	3524578	off
Romeo	32	on
Mercutio	32	on

Output

```
1 1 2 3 5 8 13 21 34 55 89 144
233 377 610 987 1597 2584 4181
6765 10946 17711 28657 46368
75025 121393 196418 317811
514229 832040 1346269 2178309
3524578
```

Example: Fibonacci sequence

```
              Scene IV: The finale.

Mercutio:  Are you better than me?  You
  bastard.

[Exit Mercutio]
[Enter Juliet]

Romeo:  You are my pretty rose!

Juliet:  You coward!  You are as bad as
  Mercutio.  Recall my final goodbye.

[Exit Juliet]

Romeo:  Am I as cursed as a damned hound?
```

Variables

Juliet	3524578	off
Romeo	32	on
Mercutio	32	on

Output

```
1 1 2 3 5 8 13 21 34 55 89 144
233 377 610 987 1597 2584 4181
6765 10946 17711 28657 46368
75025 121393 196418 317811
514229 832040 1346269 2178309
3524578
```

Significance

- DeCSS implementation would be protected by free speech laws

Significance

- DeCSS implementation would be protected by free speech laws
- Actual Shakespeare performance in 2007:

Other themed lanauges

- **Chef** (recipes)

Other themed lanauges

- **Chef** (recipes)
- **Taxi** (directions for a taxi driver)

Other themed lanauges

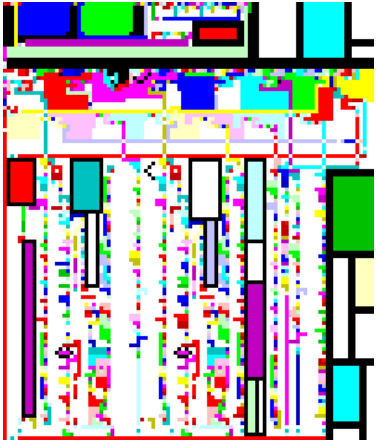
- **Chef** (recipes)
- **Taxi** (directions for a taxi driver)
- **LOLCODE** (`IM IN YR LOOP, KTHXBYE`)

Other themed lanauges

- **Chef** (recipes)
- **Taxi** (directions for a taxi driver)
- **LOLCODE** (`IM IN YR LOOP, KTHXBYE`)
- **FiM++** (*Dear Princess Celestia. . .*)

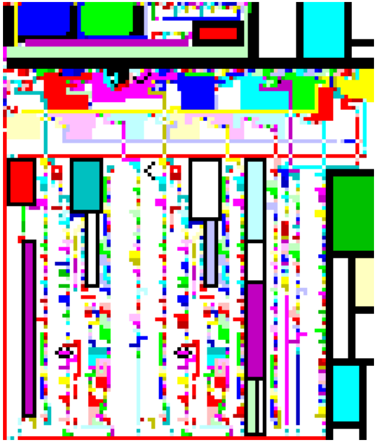
More examples

- **Piet** (abstract paintings)



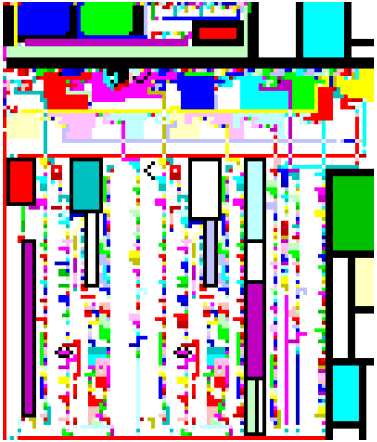
More examples

- **Piet** (abstract paintings)
- **Aceto** (follows a Hilbert curve through the source code)



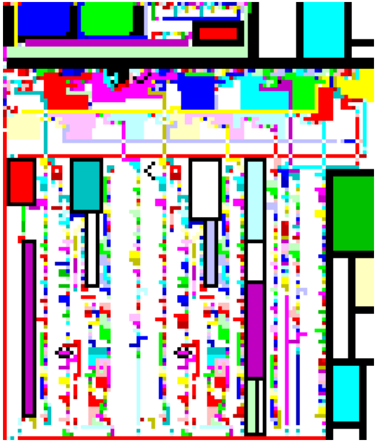
More examples

- **Piet** (abstract paintings)
- **Aceto** (follows a Hilbert curve through the source code)
- **Whitespace** (uses only space, tab, and return)



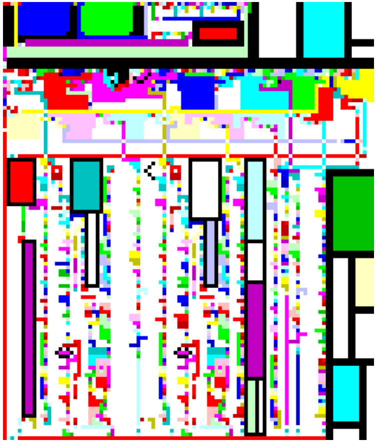
More examples

- **Piet** (abstract paintings)
- **Aceto** (follows a Hilbert curve through the source code)
- **Whitespace** (uses only space, tab, and return)
- /// (the only operation is string substitution)



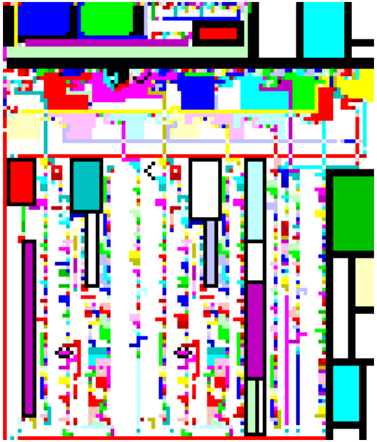
More examples

- **Piet** (abstract paintings)
- **Aceto** (follows a Hilbert curve through the source code)
- **Whitespace** (uses only space, tab, and return)
- /// (the only operation is string substitution)
- **Velato** (MIDI, tends to create jazz)



More examples

- **Piet** (abstract paintings)
- **Aceto** (follows a Hilbert curve through the source code)
- **Whitespace** (uses only space, tab, and return)
- /// (the only operation is string substitution)
- **Velato** (MIDI, tends to create jazz)



Visit `esolangs.org` for (many, many) more!

Thanks!

`sebastian@morr.cc`

`https://morr.cc`

`@blinry`

Slides and references: `morr.cc/esolangs/`