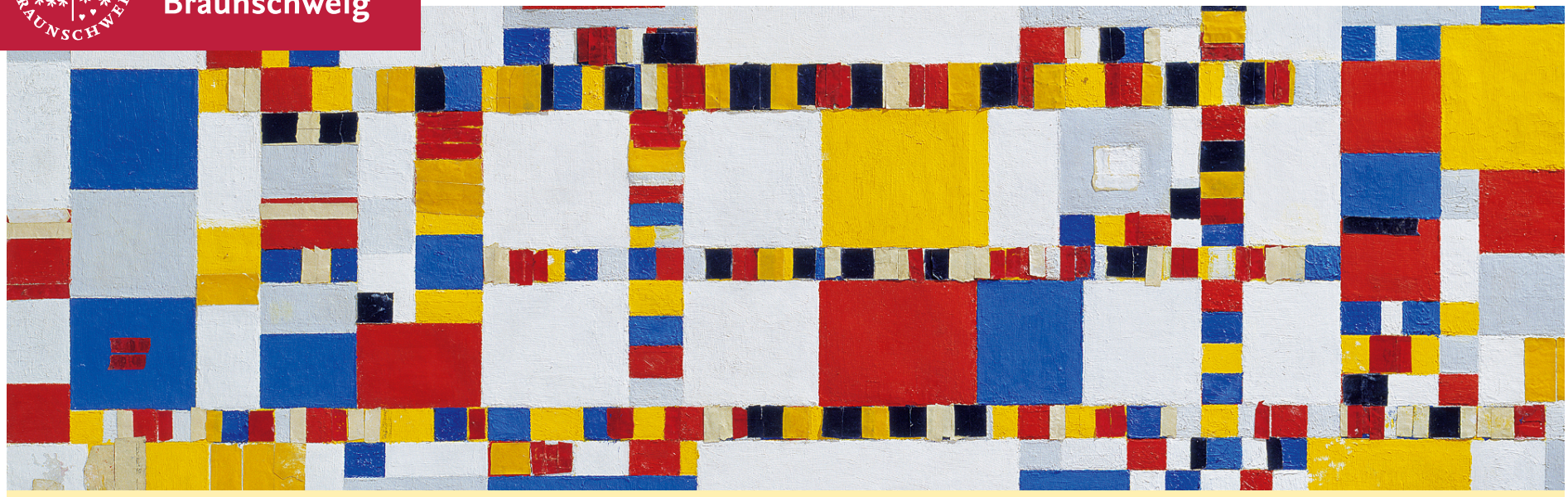
Technische Universität Braunschweig

Institut für Softwaretechnik und Fahrzeuginformatik



Esoteric Programming Languages

Seminar: Programming Languages Through the Ages

Sebastian Morr

2015–02–06

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`↩

Examples

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

Input: `f` Output: `a`↩

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

Examples

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

Input: `f` Output: `a`↩

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

Examples

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

Input: `f` Output: `a`↩

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

Examples

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

Input: `f` Output: `a`↩

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

Examples

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

Input: f Output: a↩

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

Examples

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

Input: `f` Output: `a`↩

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

Examples

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

Input: f Output: a↩

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

Examples

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

Input: `f` Output: `a`↩

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

Examples

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

Input: f Output: a↩

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

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

Examples

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

Input: `f` Output: `a`↩

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

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

Input: `h`

Examples

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

Input: `f` Output: `a`↩

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

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

Input: `h` Output: `01101000`

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:

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

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!

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
- Google released a style guide in 2007

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"'
```

Significance

- Eric Raymond released **C-INTERCAL** in 1990
- “Large”, active community
- Google released a style guide in 2007
- Donald Knuth wrote a bug report in 2010

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"'
```

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 ^
```

```
666*+.@
```

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*+.@
```

Output: 42

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

Malbolge

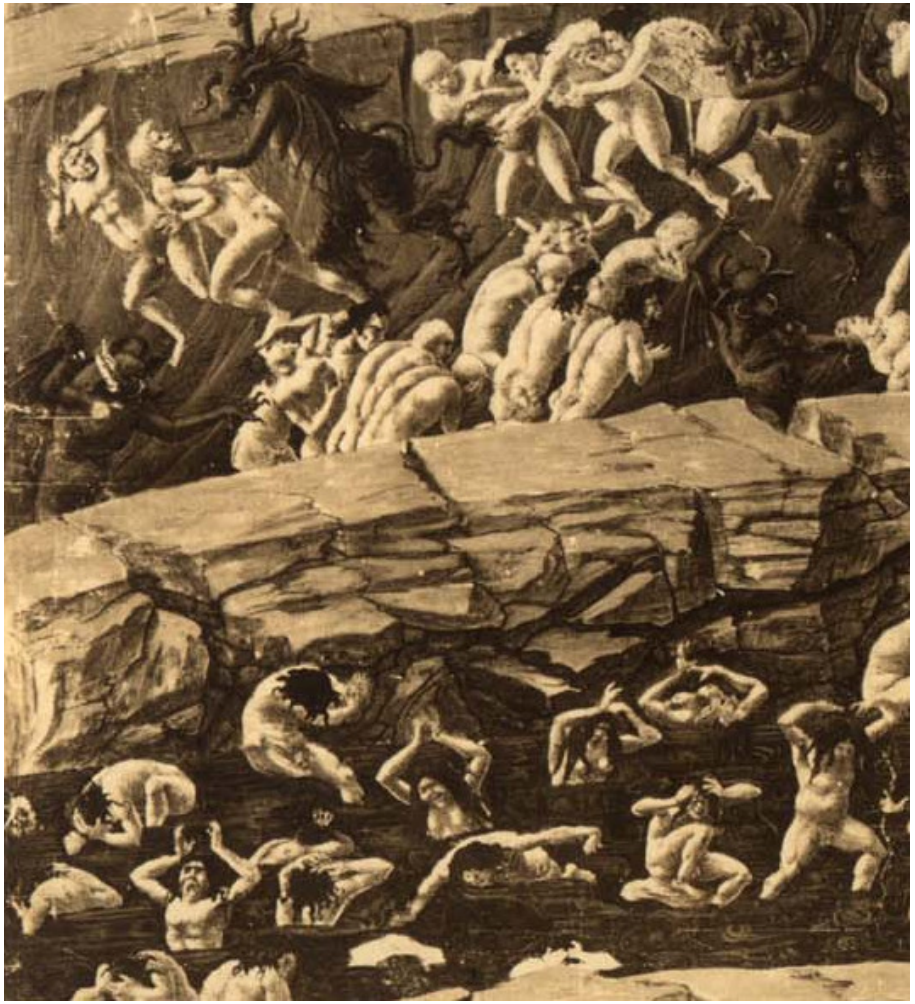
- Created in 1998 by Ben Olmstead

Malbolge

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

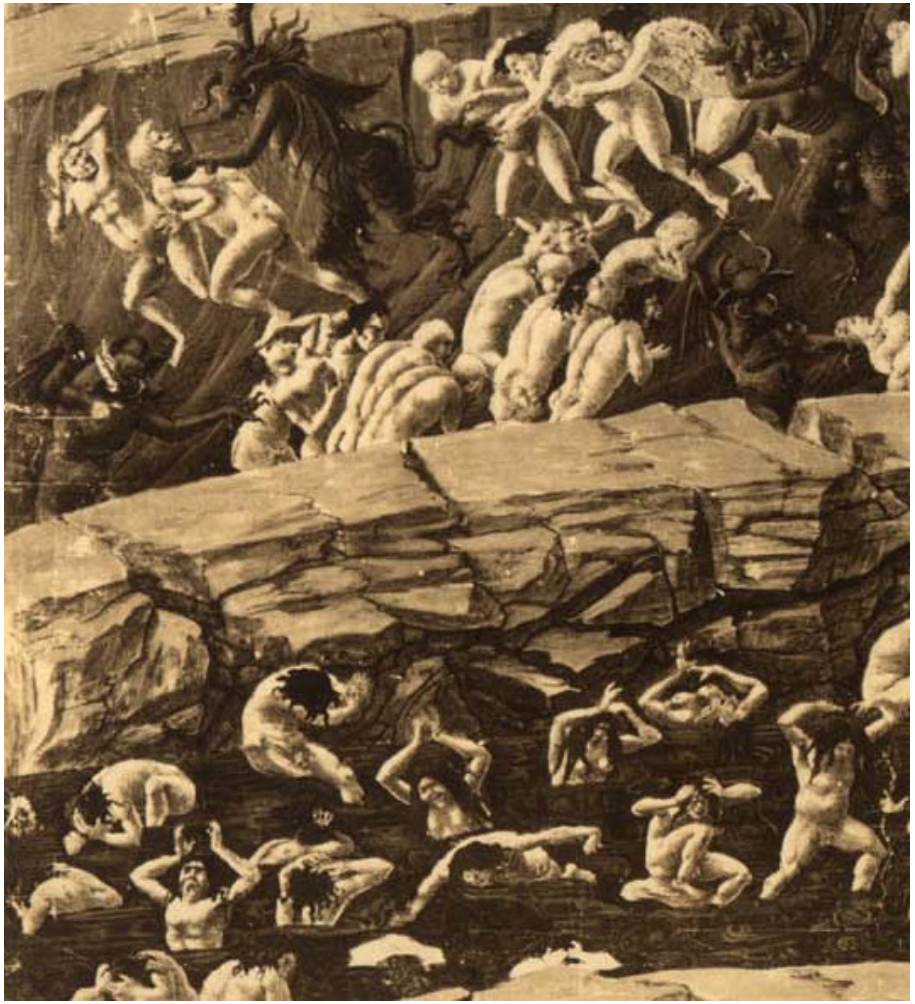
Malbolge

- Created in 1998 by Ben Olmstead
- Motivation: be incomprehensible and **hard** to use
- *Malebolge* is the eighth circle of Hell in Dante's *Inferno*



Malbolge

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



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

```
(=<‘$9]7<5YXz7wT.3,+0/o’K%$H"’~D|#z@b=‘{ˆLx8%$X
mrkpohm-kNi;gsedcba‘_ˆ]\[ZYXWVUTSRQPONMLKJIHGFE
DCBA@?>=<;:9876543s+0<oLm
```

Example: Hello world

```
(=<‘$9]7<5YXz7wT.3,+0/o’K%$H"’~D|#z@b=‘{ˆLx8%$X
mrkpohm-kNi;gsedcba‘_ˆ]\[ZYXWVUTSRQPONMLKJIHGFE
DCBA@?>=<;:9876543s+0<oLm
```

Output

```
HEllO WORld
```

Significance

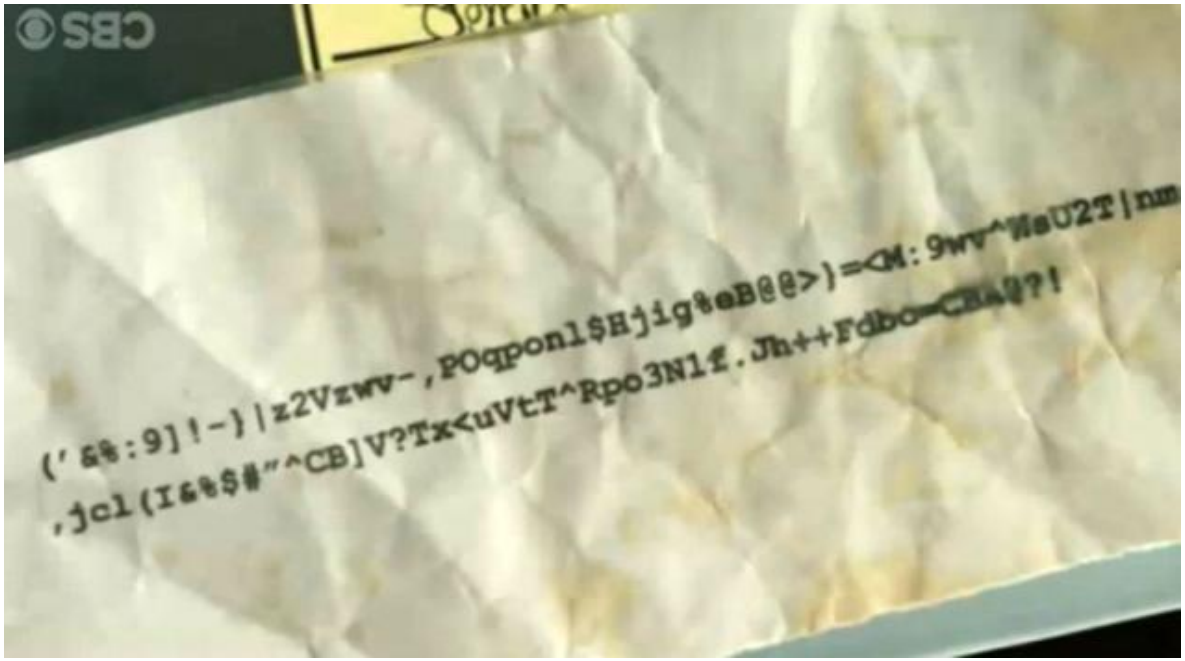
- Olmstead released “easier” **Dis**

Significance

- Olmstead released “easier” **Dis**
- Louis Scheffer uncovered some weaknesses around 2005

Significance

- Olmstead released “easier” **Dis**
- Louis Scheffer uncovered some weaknesses around 2005
- Appearance in *Elementary*:



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

Example: Powers of two

```
                    A Demonstration of Power.

Romeo, a spacy man.
Juliet, a powerful Italian lady.

          Act I: The act where it all happens.
                 Scene I: Juliet's insult.

[Enter Romeo and Juliet]

Romeo: You charming angel! Open your heart!

Juliet: You are a disgusting smelly lying rotten
  dirty pig! You are as small as the difference
  between nothing and thyself!
```

Example: Powers of two

```
                    A Demonstration of Power.

Romeo, a spacy man.
Juliet, a powerful Italian lady.

          Act I: The act where it all happens.
                 Scene I: Juliet's insult.

[Enter Romeo and Juliet]

Romeo: You charming angel! Open your heart!

Juliet: You are a disgusting smelly lying rotten
  dirty pig! You are as small as the difference
  between nothing and thyself!
```

Example: Powers of two

```
                    A Demonstration of Power.

Romeo, a spacy man.
Juliet, a powerful Italian lady.

          Act I: The act where it all happens.
                 Scene I: Juliet's insult.

[Enter Romeo and Juliet]

Romeo: You charming angel! Open your heart!

Juliet: You are a disgusting smelly lying rotten
  dirty pig! You are as small as the difference
  between nothing and thyself!
```

Example: Powers of two

```
                    A Demonstration of Power.

Romeo, a spacy man.
Juliet, a powerful Italian lady.

          Act I: The act where it all happens.
                 Scene I: Juliet's insult.

[Enter Romeo and Juliet]

Romeo: You charming angel! Open your heart!

Juliet: You are a disgusting smelly lying rotten
  dirty pig! You are as small as the difference
  between nothing and thyself!
```

Example: Powers of two

```
                    A Demonstration of Power.

Romeo, a spacy man.
Juliet, a powerful Italian lady.

          Act I: The act where it all happens.
                 Scene I: Juliet's insult.

[Enter Romeo and Juliet]

Romeo: You charming angel! Open your heart!

Juliet: You are a disgusting smelly lying rotten
  dirty pig! You are as small as the difference
  between nothing and thyself!
```

Example: Powers of two

```
                    A Demonstration of Power.

Romeo, a spacy man.
Juliet, a powerful Italian lady.

          Act I: The act where it all happens.
                 Scene I: Juliet's insult.

[Enter Romeo and Juliet]

Romeo: You charming angel! Open your heart!

Juliet: You are a disgusting smelly lying rotten
  dirty pig! You are as small as the difference
  between nothing and thyself!
```

Example: Powers of two

```
                    A Demonstration of Power.

Romeo, a spacy man.
Juliet, a powerful Italian lady.

          Act I: The act where it all happens.
                 Scene I: Juliet's insult.

[Enter Romeo and Juliet]

Romeo: You charming angel! Open your heart!

Juliet: You are a disgusting smelly lying rotten
  dirty pig! You are as small as the difference
  between nothing and thyself!
```

Example: Powers of two

```
                    A Demonstration of Power.

Romeo, a spacy man.
Juliet, a powerful Italian lady.

          Act I: The act where it all happens.
                 Scene I: Juliet's insult.

[Enter Romeo and Juliet]

Romeo: You charming angel! Open your heart!

Juliet: You are a disgusting smelly lying rotten
  dirty pig! You are as small as the difference
  between nothing and thyself!
```

Example: Powers of two

```
                    A Demonstration of Power.

Romeo, a spacy man.
Juliet, a powerful Italian lady.

          Act I: The act where it all happens.
                 Scene I: Juliet's insult.

[Enter Romeo and Juliet]

Romeo: You charming angel! Open your heart!

Juliet: You are a disgusting smelly lying rotten
  dirty pig! You are as small as the difference
  between nothing and thyself!
```

Example: Powers of two

```
                    A Demonstration of Power.

Romeo, a spacy man.
Juliet, a powerful Italian lady.

          Act I: The act where it all happens.
                 Scene I: Juliet's insult.

[Enter Romeo and Juliet]

Romeo: You charming angel! Open your heart!

Juliet: You are a disgusting smelly lying rotten
  dirty pig! You are as small as the difference
  between nothing and thyself!
```

Example: Powers of two

```
                    A Demonstration of Power.

Romeo, a spacy man.
Juliet, a powerful Italian lady.

          Act I: The act where it all happens.
                 Scene I: Juliet's insult.

[Enter Romeo and Juliet]

Romeo: You charming angel! Open your heart!

Juliet: You are a disgusting smelly lying rotten
  dirty pig! You are as small as the difference
  between nothing and thyself!
```

Example: Powers of two (cont.)

```
                    Scene II: A vicious circle.

Juliet: Speak your mind!

Romeo: You are as beautiful as the product of a
  pretty flower and thyself! Open your heart!

Juliet: Are you better than me? If so, we must
  return to scene II!
```

Example: Powers of two (cont.)

```
                    Scene II: A vicious circle.

Juliet: Speak your mind!

Romeo: You are as beautiful as the product of a
  pretty flower and thyself! Open your heart!

Juliet: Are you better than me? If so, we must
  return to scene II!
```

Example: Powers of two (cont.)

```
                Scene II: A vicious circle.

Juliet: Speak your mind!

Romeo: You are as beautiful as the product of a
  pretty flower and thyself! Open your heart!

Juliet: Are you better than me? If so, we must
  return to scene II!
```

Example: Powers of two (cont.)

```
                Scene II: A vicious circle.

Juliet: Speak your mind!

Romeo: You are as beautiful as the product of a
  pretty flower and thyself! Open your heart!

Juliet: Are you better than me? If so, we must
  return to scene II!
```

Example: Powers of two (cont.)

```
                    Scene II: A vicious circle.

Juliet: Speak your mind!

Romeo: You are as beautiful as the product of a
  pretty flower and thyself! Open your heart!

Juliet: Are you better than me? If so, we must
  return to scene II!
```

Example: Powers of two (cont.)

```
                    Scene II: A vicious circle.

Juliet: Speak your mind!

Romeo: You are as beautiful as the product of a
  pretty flower and thyself! Open your heart!

Juliet: Are you better than me? If so, we must
  return to scene II!
```

Example: Powers of two (cont.)

```
                Scene II: A vicious circle.

Juliet: Speak your mind!

Romeo: You are as beautiful as the product of a
  pretty flower and thyself! Open your heart!

Juliet: Are you better than me? If so, we must
  return to scene II!
```

Example: Powers of two (cont.)

```
                Scene II: A vicious circle.

Juliet: Speak your mind!

Romeo: You are as beautiful as the product of a
  pretty flower and thyself! Open your heart!

Juliet: Are you better than me? If so, we must
  return to scene II!
```

Output

```
2 4 8 16 32 64
```

Significance

- Proposed DeCSS implementation

Significance

- Proposed DeCSS implementation
- Actual Shakespeare performance in 2007:

Conclusion

Esoteric Programming Languages . . .

- rarely seem to go out of fashion

Conclusion

Esoteric Programming Languages . . .

- rarely seem to go out of fashion
- are playgrounds for language designers

Conclusion

Esoteric Programming Languages . . .

- rarely seem to go out of fashion
- are playgrounds for language designers
- pose interesting puzzles

Conclusion

Esoteric Programming Languages . . .

- rarely seem to go out of fashion
- are playgrounds for language designers
- pose interesting puzzles

Good Ressource: `http://esolangs.org`

Thanks!

Sebastian Morr
sebastian@morr.cc
http://morr.cc
@blinry