LOLCODE
Welcome to the last day of class!

Dr. Mayfield and Dr. Lam

Department of Computer Science
James Madison University

Dec 11, 2015
“LOLCODE is an esoteric programming language inspired by the funny things that cats say on the Internet.” [http://lolcode.org/](http://lolcode.org/)

What does *esoteric* mean?

- “Confined to and understandable by only an enlightened inner circle.” (WordNet)
- “Designed to test the boundaries of programming language design, as a proof of concept, as software art, or as a joke.” [http://en.wikipedia.org/wiki/Esoteric_programming_language](http://en.wikipedia.org/wiki/Esoteric_programming_language)
OH, HI; I'M HERE FROM THE INTERNET.
\ WHAT ARE YOU DOING!?
GLUING CAPTIONS TO YOUR CATS.

http://xkcd.com/262/
First program

HAI 1.2
    BTW prints a greeting
    VISIBLE "HAI WORLD!!!1!"
KTHXBYE

OBTW
    This program doesn't do too much and you may be wondering why it needs such a long comment, well, it's to help you understand how multiple line comments work!

TLDR
HAI 1.2
    VISIBLE "O RLY?"
KTHXBYE
HAI 1.2
BTW how to declare variables
I HAS A foo
I HAS A bar
BTW how to assign variables
foo R 1
bar R 2.34
BTW initialization syntax
I HAS A baz ITZ "OMG!"
KTHXBYE

HAI 1.2
I HAS A foo ITZ 1
BTW type casting
MAEK foo A YARN
VISIBLE foo
KTHXBYE

HAI 1.2
I HAS A foo ITZ 1
BTW another way
foo IS NOW A YARN
VISIBLE foo
KTHXBYE
Types and values

Data types
- NUMBR = integer
- NUMBAR = decimal
- YARN = string
- NOOB = nil (no value)

Boolean (TROOF)
- WIN = true
- FAIL = false

Concatenation
- SMOOSH x AN y AN z MKAY

Special characters
- ":)
- ":>
- ":o
- "::
- ":::

Standard I/O
- VISIBLE <expression>
- Can’t format output
- GIMMEH <variable>
- Works like getline
- Can’t parse strings
Math and logic

**SUM OF** `<x>` **AN** `<y>` **BTW** `+`
**DIFF OF** `<x>` **AN** `<y>` **BTW** `−`
**PRODUKT OF** `<x>` **AN** `<y>` **BTW** `∗`
**QUOSHUNT OF** `<x>` **AN** `<y>` **BTW** `/`

**MOD OF** `<x>` **AN** `<y>` **BTW** modulo
**BIGGR OF** `<x>` **AN** `<y>` **BTW** max
**SMALLR OF** `<x>` **AN** `<y>` **BTW** min

Note: `<x>` and `<y>` may each be expressions in the above, so mathematical operators can be nested and grouped indefinitely.

**BOTH OF** `<x>` **[AN]** `<y>` **BTW** and: WIN iff `x=WIN`, `y=WIN`
**EITHER OF** `<x>` **[AN]** `<y>` **BTW** or: FAIL iff `x=FAIL`, `y=FAIL`
**WON OF** `<x>` **[AN]** `<y>` **BTW** xor: FAIL if `x=y`
**NOT** `<x>` **BTW** unary negation: WIN if `x=FAIL`

**ALL OF** `<x>` **[AN]** `<y>` **... MKAY** BTW infinite arity AND
**ANY OF** `<x>` **[AN]** `<y>` **... MKAY** BTW infinite arity OR
Comparison

BOTH SAEM \langle x \rangle \ [AN] \langle y \rangle \quad BTW WIN \ iff \ x == y
DIFFPRINT \langle x \rangle \ [AN] \langle y \rangle \quad BTW WIN \ iff \ x != y

BOTH SAEM \langle x \rangle \ AN \ BIGGR \ OF \ \langle x \rangle \ AN \ \langle y \rangle \quad BTW \ x \ >= \ y
BOTH SAEM \langle x \rangle \ AN \ SMALLR \ OF \ \langle x \rangle \ AN \ \langle y \rangle \quad BTW \ x \ <= \ y
DIFFPRINT \langle x \rangle \ AN \ BIGGR \ OF \ \langle x \rangle \ AN \ \langle y \rangle \quad BTW \ x \ > \ y
DIFFPRINT \langle x \rangle \ AN \ SMALLR \ OF \ \langle x \rangle \ AN \ \langle y \rangle \quad BTW \ x \ < \ y

- The final value of an expression statement is placed in the temporary variable IT.
- IT’s value remains in local scope and exists until the next time it is replaced with a bare expression.
- Condition statements use the IT variable implicitly.
If statements

Both saem animal an "cat"
O rly?
   Ya rly
      visible "joo hav a cat"
   no wai
      visible "joo sux"
oic

Else-if example:

Both saem animal an "cat", o rly?
   ya rly, visible "joo hav a cat"
   mebbe both saem animal an "maus"
      visible "nom nom nom. i eated it."
oic

- Note the comma operator makes code more compact
Case statements

COLOR, WTF?
  OMG "R"
    VISIBLE "RED FISH"
  GTFO
OMG "Y"
  VISIBLE "YELLOW FISH"
OMG "G"
OMG "B"
  VISIBLE "FISH HAS A FLAVOR"
  GTFO
OMGWTF
  VISIBLE "FISH IS TRANSPARENT"
OIC

➤ As opposed to switch, case, break, and default
For/while loops

Loop syntax

```
IM IN YR <label> <operation> YR <variable>
[TIL|WILE <expression>]
 <code block>
IM OUTTA YR <label>
```

The operation can be **UPPIN** (increment by one), **NERFIN** (decrement by one), or any unary function.

For example

```
IM IN YR LOOP UPPIN YR VAR TIL BOTH SAEM VAR AN 10
  VISIBLE SUM OF VAR AN 1
IM OUTTA YR LOOP
```
Getting started

Useful reading

- Background: http://en.wikipedia.org/wiki/LOLCODE
- Specification: https://github.com/justinmeza/lolcode-spec

Web based interpreter

- http://asgaard.co.uk/misc/loljs/
- Includes nifty examples and cheat sheet
- Source code: https://code.google.com/p/loljs/

Command line

- lci main.lol  
  
  click here for template files