Telling a story with your code

Literate Programming with Notebooks

Melissa Weber Mendonça

RSE2017

UFSC

Motivation

Logo by winner Matt Zucker

The International Obfuscated C Code Contest

[The judges | **IOCCC home page** | How to enter | FAQ | Mirrors |
IOCCC news | People who have won | Winning entries]

The source code for the winners of the 24th IOCCC has been released.

Please see the following news items.

Goals of the Contest

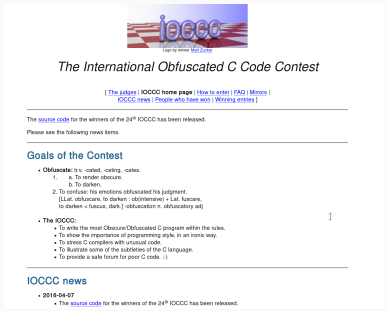
- **Obfuscate:** tr.v. -cated, -cating, -cates.
  1. a. To render obscure.
     b. To darken.
  2. To confuse: his emotions obfuscated his judgment.

  [LLat. obfuscare, to darken : ob(intensive) + Lat. fuscare,
  to darken < fuscus, dark.] -obfuscation n. obfuscatory adj

- **The IOCCC:**
  - To write the most Obscure/Obfuscated C program within the rules.
  - To show the importance of programming style, in an ironic way.
  - To stress C compilers with unusual code.
  - To illustrate some of the subtleties of the C language.
  - To provide a safe forum for poor C code. :-)

IOCCC news

- **2016-04-07**
  - The source code for the winners of the 24th IOCCC has been released.



The International Obfuscated C Code Contest

[The judges | **IOCCC home page** | How to enter | FAQ | Mirrors | IOCCC news | People who have won | Winning entries]

The source code for the winners of the 24th IOCCC has been released.

Please see the following news items.

Goals of the Contest

- **Obfuscate:** tr.v. -cated, -cating, -cates.
  1. a. To render obscure.
     b. To darken.
  2. To confuse: his emotions obfuscated his judgment.
     [Lat. obfuscare, to darken : ob(intensive) + Lat. fuscare, to darken < fuscus, dark.] -obfuscation n. obfuscatory adj
- **The IOCCC:**
  - To write the most Obscure/Obfuscated C program within the rules.
  - To show the importance of programming style, in an ironic way.
  - To stress C compilers with unusual code.
  - To illustrate some of the subtleties of the C language.
  - To provide a safe forum for poor C code. :-)

IOCCC news

- **2016-04-07**
  - The source code for the winners of the 24th IOCCC has been released.

Our main motivation is something like this. I don't know if you've heard of this contest. Basically, people are competing to see who writes the most obfuscated code, the less legible one.

Motivation

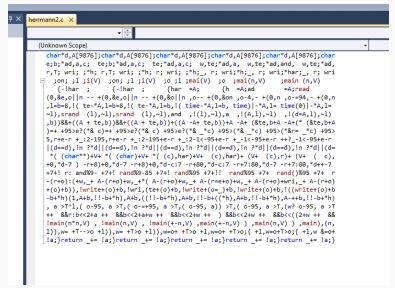
herrmann2.c

(Unknown Scope)

```
char*d,A[9876];char*d,A[9876];char*d,A[9876];char*d,A[9876];char*d,A[9876];char
e;b;*ad,a,c;  te;b;*ad,a,c;  te;*ad,a,c;  w,te;*ad,a,  w,te;*ad,and,  w,te;*ad,
r,T; wri; ;*h; r,T; wri; ;*h; r; wri; ;*h;_, r; wri;*h;_, r; wri;*har;_, r; wri
  ;on; ;l ;i(V)  ;on; ;l ;i(V)  ;o ;l ;mai(V)  ;o  ;mai(n,V)    ;main (n,V)
   {-!har  ;      {-!har  ;      {har  =A;      {h  =A;ad        =A;read
(0,&e,o||n -- +(0,&e,o||n -- +(0,&o||n ,o-- +(0,&on ,o-4,- +(0,n ,o-=94,- +(0,n
,l=b=8,!( te-*A,l=b=8,!( te-*A,l=b,!( time-*A,l=b, time)|-*A,l= time(0)|-*A,l=
~l),srand  (l),~l),srand  (l),~l),and  ,!(l),~l),a  ,!(A,l),~l)  ,!(d=A,l),~l)
,b))&&+((A + te,b))&&+((A + te,b))+((A -A+ te,b))+A -A+ (&te,b+A -A+(* (&te,b+A
)=+ +95>e?(*& c)=+ +95>e?(*& c) +95>e?(*& _*c) +95>(*& _*c) +95>(*&r= _*c) +95>
5,r+e-r +_:2-195,r+e-r +_:2-195+e-r +_:2-1<-95+e-r +_-1<-95+e-r ++?_-1<-95+e-r
|(d==d),!n ?*d||(d==d),!n ?*d||(d==d),!n ?*d||(d==d),!n ?*d||(d==d),!n ?*d||(d=
 *( (char**)+V+ *( (char)+V+ *( (c),har)+V+  (c),har)+ (V+  (c),r)+ (V+  (  c),
+0,*d-7 ) -r+8)+0,*d-7 -r+8)+0,*d-c:7 -r+80,*d-c:7 -r+7:80,*d-7 -r+7:80,*d++-7
+7+! r: and%9- +7+! rand%9-85 +7+! rand%95 +7+!!  rand%95 +7+  rand()%95 +7+  r
-(r+o):(+w,_+ A-(r+o)+w,_+*( A-(r+o)+w,_+ A-(r=e+o)+w,_+ A-(r+o)+wri,_+ A-(r+o)
+(o)+b)),!write+(o)+b,!wri,(te+(o)+b,!write+(o=_)+b,!write+(o)+b,!((write+(o)+b
-b+*h)(1,A+b,!!-b+*h),A+b,((!!-b+*h),A+b,!!-b+((*h),A+b,!!-b+*h),A-++b,!!-b+*h)
, a >T^l,( o-95, a >T,( o-=+95, a >T,( o-95, a)) >T,( o-95, a >T,(w? o-95, a >T
++  &&r:b<<2+a ++  &&b<<2+a+w ++  &&b<<2+w ++  ) &&b<<2+w ++  &&b<<((2+w ++  &&
!main(n*n,V) , !main(n,V) , !main(+-n,V) ,main(+-n,V) ) ,main(n,V) ) ,main),(n,
l)),w= +T-->o +l)),w= +T>o +l)),w=o+ +T>o +l,w=o+ +T>o;{ +l,w=o+T>o;{ +l,w &=o+
!a;}return _+= !a;}return _+= !a;}return _+= !a;}return _+= !a;}return _+= !a;}
```

Telling a story with your code

Motivation



Unfortunately, most of the code we (at least for me) are used to reading is something that could be in this contest, in whatever language (especially MATLAB). It is easy for researchers and students to copy and paste code from everywhere around the internet or from legacy projects and never question how or WHY they work. My PhD advisor, Philippe Toint, used to say that code should be written and read like liteature; that you can learn how to write good code by reading good code. That a program should ideally be read like a scientific article. It took me a long time to really understand what he meant, but when I read about literate programming, I knew this is what he was talking about.

Literate Programming

Donald Knuth, Literate Programming (1984)

"Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.

The practitioner of literate programming can be regarded as an essayist, whose main concern is with exposition and excellence of style. Such an author, with thesaurus in hand, chooses the names of variables carefully and explains what each variable means."

Don Knuth

Telling a story with your code

Literate Programming

Literate Programming

Donald Knuth, Literate Programming (1984)

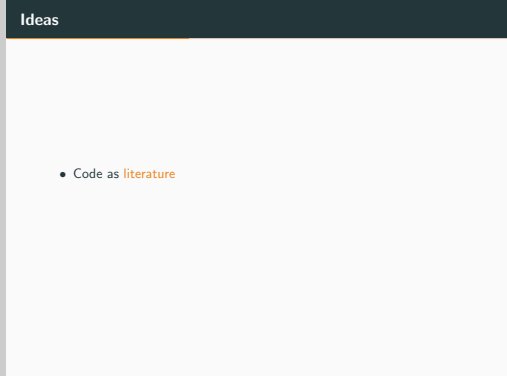
"Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.
The practitioner of literate programming can be regarded as an essayist, whose main concern is with exposition and excellence of style. Such an author, with thesaurus in hand, chooses the names of variables carefully and explains what each variable means."

Don Knuth

Enter Don Knuth. I'm sure you've heard of him before. In 1981 he thought about these things because he disagreed that structured programming was necessarily the way to go forward. In fact, he thought programming following the computer logic was a bad way to write code. He created the idead of Literate Programming based on this idea: "Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do."

Ideas

- Code as literature

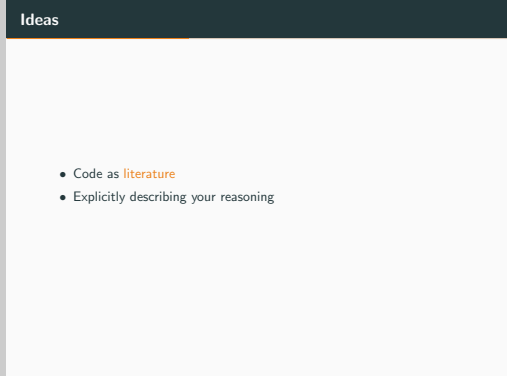


So there are some interesting points we can make.

- The first idea is to recognize "code as literature". Not only should we be able to read good code and learn from it, our code should tell a story: we should describe not only how we are solving the problem at hand, but we should describe why we are solving it that way.
- This is the second point: "Explicitly describing your reasoning". Someone reading our code should be able to detect why we chose this implementation, why this is the appropriate method, why did we choose a while instead of choosing a for. After all, if your code is reasonable, it should not be so difficult to understand what is being done.
- But more than this, the idea of literate programming is also to provide an organic documentation, meaning that code and documentation are linked in such a way that it's impossible to lose compatibility or to change the code without changing the documentation.

Ideas

- Code as literature
- Explicitly describing your reasoning



So there are some interesting points we can make.

- The first idea is to recognize "code as literature". Not only should we be able to read good code and learn from it, our code should tell a story: we should describe not only how we are solving the problem at hand, but we should describe why we are solving it that way.
- This is the second point: "Explicitly describing your reasoning". Someone reading our code should be able to detect why we chose this implementation, why this is the appropriate method, why did we choose a while instead of choosing a for. After all, if your code is reasonable, it should not be so difficult to understand what is being done.
- But more than this, the idea of literate programming is also to provide an organic documentation, meaning that code and documentation are linked in such a way that it's impossible to lose compatibility or to change the code without changing the documentation.

Ideas

- Code as literature
- Explicitly describing your reasoning
- Better documentation

Ideas

- Code as literature
- Explicitly describing your reasoning
- Better documentation

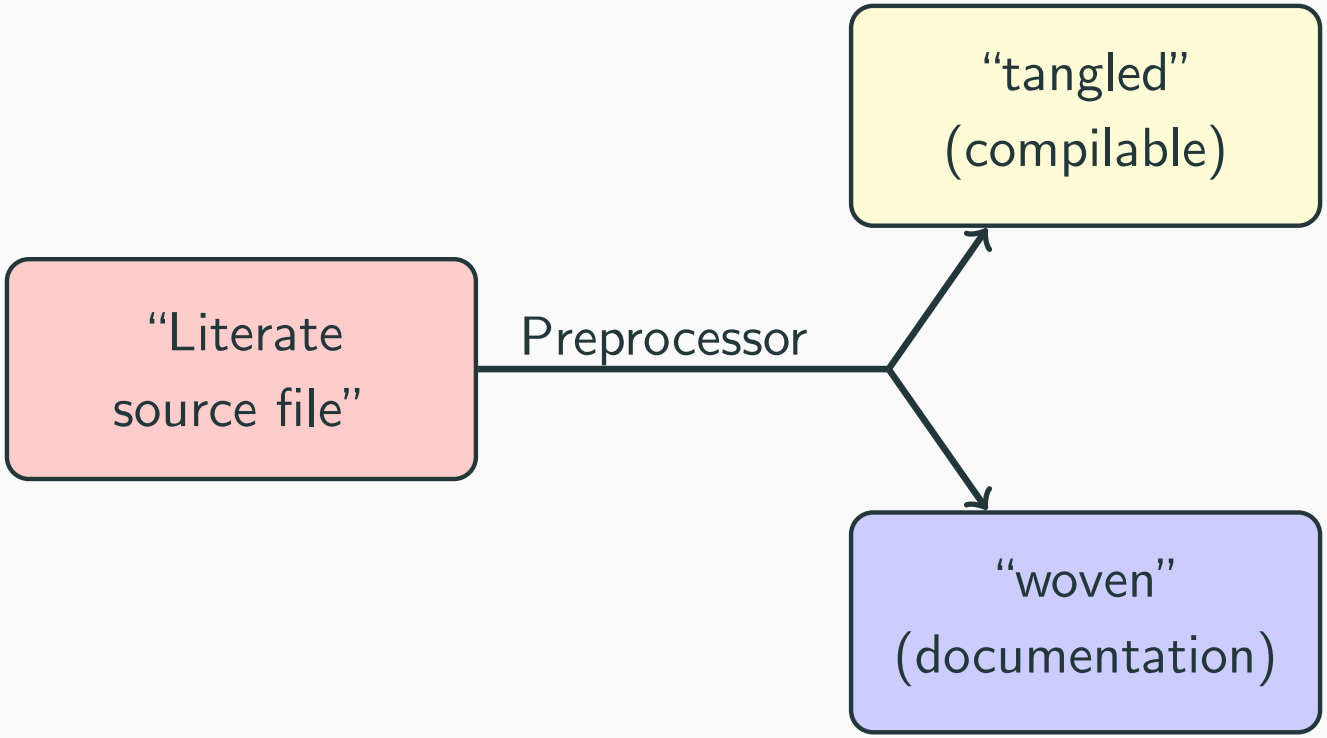
So there are some interesting points we can make.

- The first idea is to recognize "code as literature". Not only should we be able to read good code and learn from it, our code should tell a story: we should describe not only how we are solving the problem at hand, but we should describe why we are solving it that way.
- This is the second point: "Explicitly describing your reasoning". Someone reading our code should be able to detect why we chose this implementation, why this is the appropriate method, why did we choose a while instead of choosing a for. After all, if your code is reasonable, it should not be so difficult to understand what is being done.
- But more than this, the idea of literate programming is also to provide an organic documentation, meaning that code and documentation are linked in such a way that it's impossible to lose compatibility or to change the code without changing the documentation.

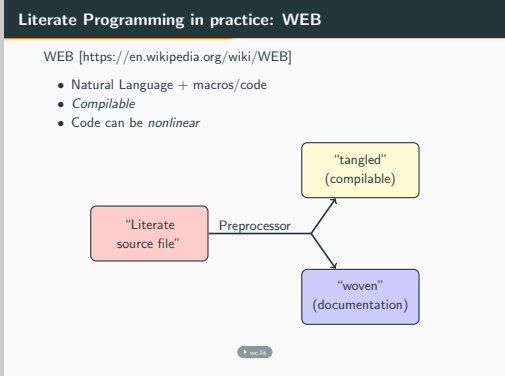
Literate Programming in practice: WEB

WEB [https://en.wikipedia.org/wiki/WEB]

- Natural Language + macros/code
- *Compilable*
- Code can be *nonlinear*



wc.lit



web is a combination of two languages: a Document formatting Language and Programming language. Initially, the idea was to use Pascal and TeX, but there are other options. The main characteristic is to mix natural language with code and or macros, that there is only one final compilable document, and code can be written in a nonlinear way, that is, we can cross reference other sections or procedures in the document so that the document does not have to be written in the order the machine expects.

Other ideas for implementation

- WEB — Pascal + TEX (Knuth)
- CWEB — C + TEX (Knuth and Silvio Levy)
- Sweave — R + TEX
- knitr — R+TEX
- noweb — Language? + Format? (HTML)
- iJulia
- Pweave — Python + Format? (Pandoc)
- iPython/Notebooks

Other ideas for implementation

- WEB — Pascal + TEX (Knuth)
- CWEB — C + TEX (Knuth and Silvio Levy)
- Sweave — R + TEX
- knitr — R+TEX
- noweb — Language? + Format? (HTML)
- iJulia
- Pweave — Python + Format? (Pandoc)
- iPython/Notebooks

There are several tools that can do literate programming.

What is not Literate Programming

- Well documented code
- Reports mixing code and text
- Automatic documentation
- Notebooks written in a “traditional” format (pasting code into a notebook instead of using an IDE)

What is *not* Literate Programming

- Well documented code
- Reports mixing code and text
- Automatic documentation
- Notebooks written in a "traditional" format (pasting code into a notebook instead of using an IDE)

Just writing good documentation or doing reports mixing code and text are not examples of LP. Just having a bunch of text and a bunch of code does not necessarily link them, and does not necessarily help explain why those choices were made, or why the code worksthe way it does.

How can this help us?

1. Communication
    - Research
    - Teaching

How can this help us?

1. Communication
   - Research
   - Teaching

Explain how we got here; which mistakes were made, bad assumptions, bad methods, WHY

How can this help us?

1. Communication
    - Research
    - Teaching
2. Research and reproducibility: how did we get here?
    - Provenance
    - Scientific workflow

Explain how we got here; which mistakes were made, bad assumptions, bad methods, WHY

Examples

`IDEB.ipynb`

A few useful tools:

- LaTeX; Pandoc
- nbextensions: codefolding, initialization cell, limit output
- Preprocessors and templates for notebooks: automatic documentation

IDEB.ipynb

A few useful tools:

- LaTeX; Pandoc
- nbextensions: codefolding, initialization cell, limit output
- Preprocessors and templates for notebooks: automatic documentation

Open example

Let's talk about it!

github.com/melissawm

github.com/melissawm/rse2017

www.mtm.ufsc.br/~melissa

melissawm@gmail.com

Telling a story with your code

2017-09-04

Let's talk about it!

github.com/melissawm
github.com/melissawm/rse2017
www.mtm.ufsc.br/~melissa
melissawm@gmail.com

Thank you.

Supporting Material i

Fernando Pérez's blog post on Literate Computing: from 2013! [link]

- Good notebooks:
  - A Crash Course in Python for Scientists (Rick Muller)
  - matplotlib — 2D and 3D plotting in Python (J.R. Johansson)
- Bad notebooks:
  - iTorch Demo (??)
  - Star Wars API