Let’s Wham!
Let’s quote!

An idiot with a computer is a faster, better idiot.
—Rich Julius

Source: http://www.jokebuddha.com/Dijkstra/recent#ixzz4StXmlURi
Let’s sing, too!

- **Jingle bells:**
  - [http://www.41051.com/xmaslyrics/jingle.html](http://www.41051.com/xmaslyrics/jingle.html)

- **Santa Claus is coming to town:**
  - [http://www.41051.com/xmaslyrics/santatown.html](http://www.41051.com/xmaslyrics/santatown.html)

- **We wish you a merry x-mas:**
  - [http://www.41051.com/xmaslyrics/wewishu.html](http://www.41051.com/xmaslyrics/wewishu.html)

- **White x-mas**
Dashing through the snow
On a one horse open sleigh
O'er the fields we go,
Laughing all the way
Bells on bob tail ring,
making spirits bright
What fun it is to laugh and sing
A sleighing song tonight

Oh, jingle bells, jingle bells
Jingle all the way
Oh, what fun it is to ride
In a one horse open sleigh

A day or two ago,
I thought I'd take a ride,
And soon Miss Fanny Bright
Was seated by my side;
The horse was lean and lank
Misfortune seemed his lot
We got into a drifted bank,
And then we got upsot.

Oh, jingle bells, jingle bells
Jingle all the way
Oh, what fun it is to ride
In a one horse open sleigh

Oh, jingle bells, jingle bells
Jingle all the way
Oh, what fun it is to ride
In a one horse open sleigh

Oh, jingle bells, jingle bells
Jingle all the way
Oh, what fun it is to ride
In a one horse open sleigh
Part I:  
A bit of language bashing
The C Programming Language - A language which combines the flexibility of assembly language with the power of assembly language.

Source: http://www.jokebuddha.com/Dijkstra/recent#ixzz4StYRnJUP
C makes it easy to shoot yourself in the foot. C++ makes it harder, but when you do, it blows away your whole leg.

—Bjarne Stroustrup

Source: http://www.jokebuddha.com/Dijkstra/recent#ixzz4StZlaRgx
For every bad story there is a good one. Recently Haskell was used in an experiment here at Yale in the Medical School. It was used to replace a C program that controlled a heart-lung machine. In the six months that it was in operation, the hospital estimates that probably a dozen lives were saved because the program was far more robust than the C program, which often crashed and killed the patients.

—Paul Hudak
> By the way seriously: how do you debug Haskell programs?

I don't. If the programme doesn't work the way I intended it, I rewrite bits in a simpler fashion until I understand it well enough and it works. If you need to debug a programme [...] , what it’s telling you is that you've written it the wrong way!

—Jón Fairbairn
So obviously, it is impossible to bash Haskell.
The primary purpose of the DATA statement is to give names to constants; instead of referring to pi as 3.141592653589793 at every appearance, the variable pi can be given that value with a DATA statement and used instead of the longer form of the constant. This also simplifies modifying the program, should the value of pi change.

—FORTRAN manual for Xerox computers

Source: http://www.jokebuddha.com/Dijkstra/recent#ixzz4StbgzWr7
Shouldn’t we do some Java bashing also?
Pascal - A programming language named after a man who would turn over in his grave if he knew about it.

Source: http://www.jokebuddha.com/Dijkstra/recent#ixzz4StYfEf1d
The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offense.
—EWD
It is practically impossible to teach good programming style to students that [sic] have had prior exposure to BASIC; as potential programmers they are mentally mutilated beyond hope of regeneration.

—EWD
Let’s sing, too!

• Jingle bells:
  • http://www.41051.com/xmaslyrics/jingle.html

• Santa Claus is coming to town:
  • http://www.41051.com/xmaslyrics/santatown.html

• We wish you a merry x-mas:
  • http://www.41051.com/xmaslyrics/wewishu.html

• White x-mas
You better watch out
You better not cry
Better not pout
I'm telling you why
**Santa Claus is coming to town**

He's making a list
And checking it twice
Gonna find out Who's naughty and nice
Santa Claus is coming to town

He sees you when you're sleeping
He knows when you're awake
He knows if you've been bad or good
So be good for goodness sake!

O! You better watch out!
You better not cry
Better not pout
I'm telling you why
Santa Claus is coming to town
Santa Claus is coming to town
Part II: 
Who is EWD?
Edsger Wybe Dijkstra (1930—2002)

Programmer at Mathematisch Centrum (Amsterdam).
Professor of mathematics at the Eindhoven University of Technology.
Research fellow at the Burroughs Corporation.
Schlumberger Centennial Chair in Computer Sciences, University of Texas at Austin.
http://amturing.acm.org/

(1973)
Bachman, Charles William

(1972)
Dijkstra, Edsger Wybe *

(1971)
McCarthy, John *

(1970)
Wilkinson, James Hardy ("Jim") *

(1969)
Minsky, Marvin *

(1968)
Hamming, Richard W*

(1967)
Wilkes, Maurice V.*

(1966)
Perlis, Alan J *
Primer of Algol 60 Programming

Academic Press; 1962
STRUCTURED PROGRAMMING


APIC Studies in Data Processing  No. 8

Academic Press
London  New York  San Francisco
A Subsidiary of Harcourt Brace Jovanovich, Publishers
"For a long time I have wanted to write a book somewhat along the lines of this one: on the one hand I knew that programs could have a compelling and deep logical beauty, on the other hand I was forced to admit that most programs are presented in a way fit for mechanical execution but, even if of any beauty at all, totally unfit for human appreciation."

Prentice Hall, Inc.; 1976
SELECTED WRITINGS ON COMPUTING: A PERSONAL PERSPECTIVE

Edsger W. Dijkstra
Person information
- award: Turing Award, 1972

2000 – 2009

2005

2002

2001

2000
EWD is famous also for his quotes


Computer Science is no more about computers than astronomy is about telescopes.
EWD is really famous for his quotes
https://www.brainyquote.com/quotes/authors/e/edsger_dijkstra.html
EWD is famous for his manuscripts

http://www.cs.utexas.edu/~EWD/
A Case against the GO TO Statement.

by Edsger W. Dijkstra

Technological University

Eindhoven, The Netherlands

Since a number of years I am familiar with the observation that the quality of programmers is a decreasing function of the density of go to statements in the programs they produce. Later I discovered why the use of the go to statement has such disastrous effects and did I become convinced that the go to statement should be abolished from all "higher level" programming languages (i.e. everything except -perhaps- plain machine code). At that time I did not attach too much importance to this discovery; I now submit my considerations for publication because in very recent discussions in which the subject turned up, I have been urged to do so.

My first remark is that, although the programmer's activity ends when ...
Coxeter's rabbit

On p.13 of his "Introduction to Geometry", H.S.M. Coxeter invites the reader to see (and to use spontaneously) that with \( s = (a+b+c)/2 \), \( abc \) equals

\[
(0) \quad s(s-b)(s-c) + s(s-c)(s-a) + s(s-a)(s-b) - (s-a)(s-b)(s-c)
\]

**Proof**

\[
s(s-b)(s-c) + s(s-c)(s-a)
\]

\[
= \{ \text{algebra} \}
\]

\[
= s(s-c)(2s-a-b)
\]

\[
= \{ \text{definition of } s \}
\]

\[
(1) \quad s(s-c)c
\]

\[
s(s-a)(s-b) - (s-a)(s-b)(s-c)
\]

\[
= \{ \text{algebra} \}
\]

\[
(2) \quad (s-a)(s-b)c
\]

Because both expressions (1) and (2) contain a factor \( c \), so does (0); for reasons of symmetry, (0) also contains factors \( a \) and \( b \), i.e., is a multiple of \( abc \). The coefficient equals 1 — as is trivially established with, say, \( a, b, c := 2, 2, 2 \) — and thus \( abc = (0) \) has been proved.  

(End of Proof)
(0) $s(s-b)(s-c) + s(s-c)(s-a) + s(s-a)(s-b) - (s-a)(s-b)(s-c)$

**Proof**

$$s(s-b)(s-c) + s(s-c)(s-a)$$

= \{ algebra \}

$$s(s-c)(2s-a-b)$$

= \{ definition of $s$ \}

(1) $$s(s-c)c$$

$$s(s-a)(s-b) - (s-a)(s-b)(s-c)$$

= \{ algebra \}

(2) $$s(s-a)(s-b)c$$

Because both expressions (1) and (2) contain a factor $c$, so does (0); for reasons of symmetry, (0) also contains factors $a$ and $b$, i.e. is a multiple of $abc$. The coefficient equals 1 - as is trivially established with, say, $a, b, c := 2, 2, 2$ - and thus $abc = (0)$ has been proved. (End of Proof)

Nuenen, 14 April 2002

prof. dr Edsger W. Dijkstra
Plataanstraat 5
5671 AL Nuenen
The Netherlands
Quotes about EWD are famous, too.

Alan Kay at his 1997 OOPSLA Keynote
“The Computer Revolution hasn't happend yet “
https://www.youtube.com/watch?v=oKg1hTOQXoY

“I don't know how many of you have ever met Dijkstra, but you probably know that arrogance in computer science is measured in nano-Dijkstras.”
Part III:
Some bits of EWD work
In 1959 Dijkstra published in a 3-page article 'A note on two problems in connexion with graphs' the algorithm to find the shortest path in a graph between any two given nodes, now called Dijkstra's algorithm. Its impact over the next 40 years is summarised from the article of Mikkel Thorup, 'Undirected Single Source Shortest Paths with Positive Integer Weights in Linear Time' (1999): "Since 1959, all theoretical developments in SSSP [Single-Source Shortest Paths] for general directed and undirected graphs have been based on Dijkstra's algorithm." Dijkstra's algorithm is used in SPF, Shortest Path First, which is used in the routing protocols OSPF and IS-IS. Various modifications to Dijkstra's algorithm have been proposed by many authors using heuristics to reduce the run time of shortest path search. One of the most used heuristic algorithms is the A* search algorithm, the main goal is to reduce the run time by reducing the search space.
Predicate transformer semantics
https://en.wikipedia.org/wiki/Predicate_transformer_semantics

• Symbolic execution of statements; compute weakest precondition.

  • \( wp(\text{skip}, R) = R \)

  • \( wp(\text{abort}, R) = \text{false} \)

  • \( wp(x := E, R) = R[x \leftarrow E] \)

\[
wp(x := x - 5, x > 10) = x - 5 > 10
\]

\[\iff x > 15\]
Part IV: What did EWD say?
A convincing demonstration of correctness being impossible as long as the mechanism is regarded as a black box, our only hope lies in not regarding the mechanism as a black box.

Dijkstra (1970) "Notes On Structured Programming" (EWD249), Section 3 ("On The Reliability of Mechanisms"), p. 5.
I mean, if 10 years from now, when you are doing something quick and dirty, you suddenly visualize that I am looking over your shoulders and say to yourself "Dijkstra would not have liked this", well, that would be enough immortality for me.

Dijkstra (1995) "Introducing a course on calculi" (EWD 1213).
The required techniques of effective reasoning are pretty formal, but as long as programming is done by people that don't master them, the software crisis will remain with us and will be considered an incurable disease. And you know what incurable diseases do: they invite the quacks and charlatans in, who in this case take the form of Software Engineering gurus.

Dijkstra (2000) "Answers to questions from students of Software Engineering" (EWD 1305).
There are very different programming styles. I tend to see them as Mozart versus Beethoven. When Mozart started to write, the composition was finished. He wrote the manuscript and it was 'aus einem Guss' (from one cast). In beautiful handwriting, too. Beethoven was a doubter and a struggler who started writing before he finished the composition and then glued corrections onto the page. In one place he did this nine times. When they peeled them, the last version proved identical to the first one.

Dijkstra (2001) Source: Denken als discipline, a program from Dutch public TV broadcaster VPRO from April 10th, 2001 about Dijkstra
Our intellectual powers are rather geared to master static relations and that our powers to visualize processes evolving in time are relatively poorly developed. For that reason we should do (as wise programmers aware of our limitations) our utmost to shorten the conceptual gap between the static program and the dynamic process, to make the correspondence between the program (spread out in text space) and the process (spread out in time) as trivial as possible.

It is not the task of the University to offer what society asks for, but to give what society needs.

Dijkstra (2000), "Answers to questions from students of Software Engineering" (EWD 1305).
Let’s sing, too!

- **Jingle bells:**

- **Santa Claus is coming to town:**

- **We wish you a merry x-mas:**

- **White x-mas**
We wish you a **Merry Christmas**,  
We wish you a Merry Christmas,  
We wish you a Merry Christmas,  
And a Happy New Year.  

Good tidings to you,  
And all of your kin,  
Good tidings for Christmas,  
And a Happy New Year.  

We all know that Santa's coming,  
We all know that Santa's coming,  
We all know that Santa's coming,  
And soon will be here.
A short ad is due.

Ein Film von Mark Widiger.
Edsger Wybe Dijkstra
(1930—2002)

Part V:
What to put into EWD’s words?

The Humble Programmer (1972)
FORTRAN's tragic fate has been its wide acceptance, mentally chaining thousands and thousands of programmers to our past mistakes.
Javascript's tragic fate has been its wide acceptance, mentally chaining millions and millions of programmers to severe forms of language disorders.
If you want more effective programmers, you will discover that they should not waste their time debugging, they should not introduce the bugs to start with.
If you want more effective programmers, you will discover that they should not waste their time on pair programming or scrums, they should just know what they are doing, once they are told what to do.
The effective exploitation of his powers of abstraction must be regarded as one of the most vital activities of a competent programmer.
There is no program small enough that couldn’t be made more insightful by throwing in a few folds, maps, type classes, fixed-point operators, monoids, monads, functors, and other such mambo jambo.
When FORTRAN has been called an infantile disorder, full PL/1, with its growth characteristics of a dangerous tumor, could turn out to be a fatal disease.
When Cobol has been called a mortally ill language for many decades, Prolog, with its complete lack of perceived coolness, has really stopped breathing.
LISP has been jokingly described as "the most intelligent way to misuse a computer". I think that description is a great compliment because it transmits the full flavor of liberation: LISP has assisted a number of our most gifted fellow humans in thinking previously impossible thoughts.
Haskell has been jokingly described as "the most type-safe way to misuse a computer". I think that description is a great compliment: Haskell prevents people from testing or running programs; they are entirely satisfied with just type-checking them.
Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence.
Program testing is a sign of missing self-confidence combined with admission of mediocre skills—missing the important ability to derive programs that are correct by construction.
What to put into EDW’s words?

Let’s use more diverse sources.
Simplicity is prerequisite for reliability.

How do we tell truths that might hurt? (numbered EWD498, written 1975) was written as a series of aphorisms, and is the source of several popular quotations. It was also published in Selected Writings on Computing: A Personal Perspective.
NO-OP is the best program in universe.
Elegance is not a dispensable luxury but a quality that decides between success and failure.

PHP scripts are obviously not elegant, but some argue that they don’t fail too often. Thanks to Donald Trump we now have a name for this: Post-truth politics.
It is time to unmask the computing community as a Secret Society for the Creation and Preservation of Artificial Complexity.

Dijkstra (1996) "The next fifty years" (EWD 1243a).
It is time to admit that many of us benefit from Job Security technologies and fashion products the use of which in all these useless information systems consumes much of our society’s budget.
In the wake of the Cultural Revolution and now of the recession I observe a mounting pressure to co-operate and to promote "teamwork". For its anti-individualistic streak, such a drive is of course highly suspect; some people may not be so sensitive to it, but having seen the Hitlerjugend in action suffices for the rest of your life to be very wary of "team spirit". Very.

Dijkstra (1994) "The strengths of the academic enterprise" (EWD 1175).
Permitting everyone to apply agile methodologies is an immediate consequence of the right of religious freedom.
It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration.

How do we tell truths that might hurt? (numbered EWD498, written 1975) was written as a series of aphorisms, and is the source of several popular quotations. It was also published in Selected Writings on Computing: A Personal Perspective.
Since 77% of all CS students begin their studies solely on the grounds of prior PHP or JavaScript programming experience, the overall construct of an academic CS curriculum is systematically useless and the tax payers must step in here to stop this.
The question of whether Machines Can Think... is about as relevant as the question of whether Submarines Can Swim.

The question of whether Machines Can Think ... has been sufficiently studied by Westworld and other SciFi movies. However, humans do not exactly provide good role models.
May all your software languages be nice to you during Xmas and next year!
Let’s sing, too!

- Jingle bells:

- Santa Claus is coming to town:

- We wish you a merry x-mas:

- White x-mas