

## Holon programming regained

Mitchell Gerrard

### Abstract

One of the main inspirations for literate programming was a technical report entitled *Holon Programming: A Survey*, by Pierre-Arnoul de Marneffe. It was privately circulated among computer scientists in 1973. The document thereafter became a Borgesian mythical book, existing only in citations by Knuth. This article narrates the search-and-rescue mission of this rare book, and highlights a few of its innovations. The full report is available at [github.com/holon-scribe/holon-programming](https://github.com/holon-scribe/holon-programming).

### 1 Ancient history

*And indeed, he composed a fair great book with figures, but it is not printed as yet that I know of.*

—François Rabelais, *Pantagruel* (1532)

In 1973, a Belgian computer scientist named Pierre-Arnoul de Marneffe was finishing a report describing his ideal programming language. More on this later.

A few years prior, Edsger Dijkstra had circulated his *Notes on Structured Programming* [3]. These *Notes* marked a watershed in the computing community. Dijkstra urged the systematic use of now-commonplace programming constructs such as for loops, if/then/else statements, and subroutines. He also gave a method to write complex programs by starting with an abstract description and successively refining this description into smaller, more manageable chunks. But what should one call these chunks of related computation? Dijkstra called them “pearls”, regarding a program as a necklace strung from individual pearls; Donald Knuth wrote to Dijkstra: “We need another word for pearl, though; what should it be?” [7]

Prof. de Marneffe knew what the word should be. He had recently read *The Ghost in the Machine* by Arthur Koestler, in which Koestler coins the term “holon”, denoting the various “nodes on [a] hierarchic tree which behave partly as wholes or wholly as parts, according to the way you look at them.” [12] The *holon* would be the unifying concept in de Marneffe’s synthesis of Dijkstra and Koestler. In December of 1973, de Marneffe privately circulated copies of his report entitled *Holon Programming: A Survey*.

### 2 Modern times

*I try to reason, and I tell myself you’ll return.*

—Roberta Flack, *Gone Away* (1970)

The year is now 2015. I had written my first few literate programs using Norman Ramsey’s `noweb`

tool [17], and was immediately smitten by this peculiar approach to programming. So I reread Knuth’s article that introduced literate programming, to seek the font of this love potion that was on my sleeping eyelids laid. And happy day—here were breadcrumbs: “The design of WEB was influenced primarily by the pioneering work of Pierre-Arnoul de Marneffe, whose research on what he called ‘Holon Programming’ has not received the attention it deserves.” [8] This statement was accompanied by two citations, one of them a 135-page report [6]. Yet when I searched for this report online, there were no books published under this name, there were no PDF scans, there was almost nothing save for a few tantalizing passages and descriptions of this mysterious document. Most strange.

Was *Holon Programming* a fictitious entry in *The Catalog of Lost Books* [20]? But there were extracts, and de Marneffe was a real author... surely this influential report hadn’t been lost to posterity.

I wrote to Prof. de Marneffe. He replied that he indeed had a copy in his files that he would scan and send to me, but he was recovering from a long hospital stay, so would do so after feeling better. Some time passed and I did not want to trouble Prof. de Marneffe further. I then wrote to Prof. Knuth. He replied: “I think I donated my copy to Stanford’s tech reports collection, but they don’t seem to have it”, and directed me to the only known library copy that was supposedly held in Germany. Knuth also enclosed a copy of the letter that he wrote to de Marneffe in 1974. This letter contained such specific references to the report that I was almost convinced *Holon Programming* was not a fabrication. Unfortunately some doubts remained, as the letter was dated April 1st.

The full letter is reproduced below (with permission).

### 3 Knuth’s letter

April 1, 1974

Prof. Pierre-Arnoul de Marneffe  
Université de Liège  
Service d’Informatique  
Avenue des Tilleuls 59  
B-4000 Liège, Belgium

Dear Prof. de Marneffe:

Thank you very much for sending me your survey of Holon Programming. I especially enjoyed your references to the non-computer literature (Koestler, Bernard-Shaw, Shanley, Mount Vernon, etc.) since computer scientists need to avoid insularity.

Mitchell Gerrard

[doi.org/10.47397/tb/45-2/tb140gerrard-holon](https://doi.org/10.47397/tb/45-2/tb140gerrard-holon)

I believe you are making important strides toward the development of a new programming language. There still remain some unclear areas but you are obviously addressing the correct issues; the next thing to do (it seems to me) is to program several hundred examples!

For related reading I would suggest that you carefully study Ole-Johan Dahl's papers on SIMULA since his class concept is so close to the holon concept. Also I have just heard that Brian Randell of Newcastle has been working on a so-called PEARL system.

I found your report could have been improved if you had worked entirely with tree structures instead of converting to binary trees. The original tree structure is what is really relevant, and the Dewey notation for such structure is more directly suited to the operations you discuss. The binary tree is only a machine-oriented representation of the basic concept, the discussion should stay at a higher level.

Secondly, I found the report too preoccupied with details of implementation. The people by whom it is most important that this report be read are either able to visualize easily how to implement this sort of system, or else they are people who are not likely to care how it's implemented as long as it's handled sensibly. The important thing to stress is rather the conceptual issues of how holon programming differs from and improves on today's languages.

The example didn't come until page 100, while I expect most readers would have preferred to see it immediately. Since the program is almost self-explanatory, you can let it explain the language at the same time (integration of functions!).

Ideally there should be more examples of course.

The one example raises some interesting issues since the individual holons don't quite state their assumptions. In the very first holon, for example, it is not at all clear why you 'find first word starting character' instead of going right into 'find a word **etc.**' You must already have made a decision (a) that you wouldn't assume the text begins with a nonblank, (b) that there is going to be at this level an element of data representing the last-read character, (c) that the 'find a word' routine will already have its first character in hand, and (d) that there is no need to test for a message that has no words (only a full stop). As I recall when I was solving that problem, it took me a good five minutes to reach these decisions, during which I must have considered lots of alternatives. Once this step was made the rest of the program flowed naturally. My questions are: Where should we state these assumptions? Shouldn't we mention the existence of data representing the

last-read character, even though we don't want to specify its detailed structure until later?

These issues seem to arise repeatedly and I haven't a first conclusion about what we ought to do. That's why I suggest working out hundreds of examples, as being the best kind of eating to prove the pudding at this stage. On the other hand creating the holon implementation itself is equivalent to working out quite a few examples.

Thanks again for showing me your stimulating work. I myself must get on with the writing of volume 4 of my series, so I have little energy to devote to the development of languages, but I will do my best to see that other people working in the area are kept informed of what you are doing.

Sincerely,

Donald E. Knuth  
Professor

P.S. Is there a place in Belgium whose postal code is B-6700 like the Burroughs computer?


#### 4 The survey itself

*Our Perdita is found.*

— Shakespeare, *A Winter's Tale* (1611)

More time passed, revealing more false bottoms, but, eventually, kind librarians on both sides of the Atlantic arranged for that most elusive document to be sent from Hanover to Nebraska.

And so: After more than 50 years in hiding, Pierre-Arnoul de Marneffe's *Holon Programming: A Survey*, prefaced with Knuth's letter, is returned. Make its acquaintance at the address below.

 [github.com/holon-scribe/holon-programming](https://github.com/holon-scribe/holon-programming)

I second Knuth's suggestion to jump straight to the ⟨Program Example⟩ section to get a feel for what it's all about. And then, in the hypertext spirit of *Hopscotch* [2], jump around and **go to** whichever chapter titles most draw you.

In this technical report filled with out-of-the-way observations, projected language features and imagined ecosystems (a full "holon operating system")—where is the literate programming? Well, if you attire the program example's bare pseudocode phrases with a ⟨ and ⟩ on either side, the holons transform into the code sections of WEB. So let's do just that: we'll take an extract of de Marneffe's program and translate its "holons" into corresponding sections of a WEB program. This program solves a problem from section 16 of Dijkstra's *Notes*; its details aren't relevant here. Two notes on the syntax: the **etc** keyword abbreviates unambiguous prefixes, and '#′ followed by '# #' brackets low-level statements.

## 5 A holon program and its WEB twin

```

odd inversion program
  begin find first word starting character;
    repeat find a word and print correctly;
    until end of useful file
  end
find first word etc
  begin read first symbol;
    while last read symbol is a space;
    do read next symbol
  end
:
read first symbol
  begin declare lrs: character at
    odd inversion program level;
    # lrs ← RNC ##;
  end
:

```

And here are the equivalent extracts written in WEB. The `<Global variables>` section approximates how, in de Marneffe’s language, one “declares a variable and specifies the scope by naming an outer holon.” [5]

1. This program is one possible solution to the problem posed in section 16 of Dijkstra’s *Notes*.

```

program odd_inversion;
  var <Global variables 4>
  begin <Find first word starting character 2>;
    repeat <Find a word and print correctly 3>;
    until <End of useful file 13>
  end.

```

2. `<Find first word ... 2>` ≡

```

begin <Read first symbol 11>;
  while <Last read symbol is a space 12>;
  do <Read next symbol 6>
end

```

This code is used in section 1.

```

:
10. <Global variables 4> +≡
lrs: char; {last-read symbol}

```

This code is used in section 1.

11. `<Read first symbol 11>` ≡

```

begin lrs ← RNC; {read next character}
end

```

This code is used in section 2.

```

:

```

---

*RNC*: **procedure**, §15

The resemblance is uncanny. What de Marneffe did was show how a program can be written “in the order of its design”, using phrases mostly in natural language, in digestible sections of no greater than eight lines, that can be automatically “disentangled” (de Marneffe’s word) into a fully executable program. Knuth describes de Marneffe’s approach as “a way of taking a complicated program and breaking it into small parts. Then, to understand the complicated whole, what you needed is just to understand the small parts, and to understand the relationship between each part and its neighbors.” [10]

I won’t go on to give a book report of *Holon Programming*. Instead, I’ll assign further reading and then highlight a few more prefigurings of the literate programming we know today.

To rough in more of the context in which de Marneffe’s survey is, holon-like, embedded, I recommend reading Chapters 2, 3 and 5 of *The Ghost in the Machine* through the lens of Dijkstra’s *Notes*. Koestler’s arguments employ the language of computer science (via Herbert A. Simon); the metaphors and exact phrasings he uses to describe carrying out tasks closely echo those Dijkstra uses to describe fleshing out programs. They also share a fondness for pugilistic asides. In a happy coincidence, the section that de Marneffe singles out in Koestler, “How to Build a Nest”, contains three instances of the word “web” and four instances of “weaving”.

Now for the highlights.

We come across a few false friends in comparing de Marneffe’s language with Knuth’s. The **append** command refers to defining a new section, differing from the `+≡` append operation in WEB. There is an appearance of suggested macro use; but unlike Knuth’s more straightforward macros, de Marneffe’s were to parameterize holons themselves, making them more procedure-like.

Other constructs are remarkable prototypes of those we know: the **text** command defines a section of prose to “explain the reason of some design decisions”; the **change** command is like Knuth’s change file mechanism, except the entire section must be replaced; the **etc** abbreviation keyword has turned into the ‘...’ shorthand in a WEB source file; the **output** command prints the holon program as intended for human eyes to an output device, somewhat akin to weaving; and the **synthesize** command outputs the program intended for machine consumption, akin to tangling.

Most wonderful!

But hol’ on there... if all these elements were present in de Marneffe, what exactly were Knuth’s contributions?

## 6 Woven WEBS

*Nothing about WEB is really new; I have simply combined a bunch of ideas that have been in the air for a long time.*

— Donald Knuth, *Literate Programming* (1984)

*Let no one say that I have said nothing new; the arrangement of the subject is new. When we play tennis, we both play with the same ball, but one of us places it better.*

— Blaise Pascal, *Pensées* (1657?)

Knuth’s selection, rearrangement, and improvement upon ideas “in the air” was decisive. He saw through many of the irrelevant technical details in de Marneffe’s report and grasped the essence. Knuth made the subtle but crucial design decision to bring the informal prose explanations to the forefront. (The `text` command in de Marneffe never appears in examples, and seems to be regarded as a simple “comment” mechanism, despite my hyping it just now.)

The importance of colorful language, metaphors, and rephrasings cannot be understated when thinking about the unreasonable effectiveness of literate programming. “We retain only what has been dramatised by language; any other judgment is fleeting.” [1] The storytelling elements of a good literate program act as strong fixatives in our memory. And Knuth does not limit the prose to the “informal” portions; it spills over into the formal (code) portions as well. In all of Knuth’s published literate programs, he follows most macro definitions and variable declarations with some explanatory comment.

As Knuth brings informal prose to center stage, he also casts the prettyprinted code to be its costar. Whereas de Marneffe banishes the lowly code statements to hide within ugly ‘#’ and ‘##’ curtains and leaves the holon names unmarred, Knuth reverses this: he lets the (formatted) code stand on its own and brackets the section names with (less-obtrusive) delimiters. Knuth does not at all want to give short shrift to the code. The typeset Pascal in WEB’s woven output invites the reader to see how the code syncs up with its informal explanation above.

Knuth developed the first implementation of literate programming, with the `TEX` and `METAFONT` projects being the “proof in the pudding”. WEB includes a host of features unforeseen by de Marneffe, such as commands to produce camera-ready programs. Alongside these projects Knuth brought the history of literature, typography, and book design to bear on this style of programming. Beautiful fonts, typeset code blocks, cross-referencing included with each section name, tables of contents, indices, mini-indices, appendices, bibliographies, figures . . .

But we are getting far afield. Briefly stated: there was a whole lot that was new in WEB.

## 7 Old yarns

*What threads were those, oh, ye Weird Ones, that ye wove in the years foregone.*

— Herman Melville, *Pierre; or, The Ambiguities* (1852)

I’ve focused the discussion of the genesis of holon programming on the two authors de Marneffe cites the most: Dijkstra and Koestler (“The author really doesn’t know to what extent the reader can grasp the holon concept without reading Koestler’s book.” [5]). But there was one other primary influence on de Marneffe’s language. Who? Lo and behold: Knuth.

The program example used by de Marneffe to illustrate his holon language is predated by two alternate program solutions to the same problem, given by Knuth in his letters to Dijkstra and Dahl [7]. We know de Marneffe was familiar with these letters because he cites and comments upon them. In Knuth’s first program solution, we see the program only in its final, (mostly) executable, stage, but he does something remarkably close to de Marneffe. That is, Knuth composes a program out of small code sections not exceeding eight lines, each labeled with a natural language descriptor, constructed in “the order in which the decisions were made” [7], and systematically expanded and interleaved into a machine-readable form. The code sections, called “pearls” in the letter, are identified in the left margin of the “tangled” output. The second program solution is largely the same, but Knuth explicitly groups the small independent sections (here called “classes”) in the top-down order of design. These programs differ in many details from de Marneffe’s system, but the influence in language design clearly ran in both directions. And of course there are countless other contributors who helped bring about literate programming.

So I would like to express a desire to reprint some of the precursors in a slim anthology entitled *Pre-literate Programming*. This could include selections from de Marneffe; the PEARL system mentioned by Knuth in his letter to de Marneffe [18]; Dahl’s SIMULA papers [4]; Naur’s “Programming by action clusters” [14] and “Formalization in program development” [15]; Knuth on *Structured Programming* [7]; Babbage’s “On a method of expressing by signs the action of machinery” [13]; excerpts from Dijkstra’s *Notes* [3]; selections from Chapter 2 of Wilkes et al.’s book [22]; a rifacimento of Weinberg’s book into a collection of aphorisms [21]; Towster’s work [19];

portions of Jim Dunlap’s early compiler code with forty-character-long identifiers; a two-page spread exhibiting the impact typography has on program comprehension, with a non-typeset Algol program on one side and the same program typeset with executive editor for ACM Myrtle Kellington’s standards on the opposite side; Derek Oppen’s “Prettyprinting” [16]; and, as a specimen of good storytelling, something by Shirley Jackson. (We leave out the many shoots and buds of literate programming already gathered in Knuth’s early papers from [9] and “Computer Drawn Flowcharts” and pp. 229–235 in [11].)

### Acknowledgments

Thanks Don Knuth for help in locating this report and for letting the 1974 letter be reproduced. Thank you librarians. Thanks Karl Berry, Udo Wermuth, and Andreas Scherer for your feedback and suggestions. And thank you Pierre-Arnaud Frédéric Guy Donat de Marneffe (1946–2023) for creating and sharing this groundbreaking work.

### References

- [1] G. Bachelard. *The Dialectic of Duration*. Rowman & Littlefield, 2016.
- [2] J. Cortázar. *Rayuela (Hopscotch)*. Sudamericana, 1963.
- [3] O.J. Dahl, E.W. Dijkstra, C.A.R. Hoare. *Structured programming*. Academic Press Ltd., 1972. [archive.org/details/Structured\\_Programming\\_\\_Dahl\\_Dijkstra\\_Hoare](https://archive.org/details/Structured_Programming__Dahl_Dijkstra_Hoare)
- [4] O.J. Dahl, K. Nygaard. SIMULA: an ALGOL-based simulation language. *Communications of the ACM*, 9(9):671–678, 1966.
- [5] P. de Marneffe, D. Ribbens. Holon programming. In *International Computing Symposium*, A. Günther et al., ed., Amsterdam, North Holland, 1973.
- [6] P.A. de Marneffe. *Holon Programming: A Survey*. Univ. de Liège, December 1973.
- [7] D.E. Knuth. A review of “Structured Programming”. Technical Report STAN-CS-73-371, Stanford Computer Science Department, Stanford University, Stanford, CA, 1973. [i.stanford.edu/TR/CS-TR-73-371.html](https://i.stanford.edu/TR/CS-TR-73-371.html)
- [8] D.E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.
- [9] D.E. Knuth. *Literate Programming*. CSLI, 1992.
- [10] D.E. Knuth. *Digital Typography*. CSLI, 1999.
- [11] D.E. Knuth. *Selected Papers on Computer Languages*. CSLI, 2003.
- [12] A. Koestler. *The Ghost in the Machine*. Macmillan, 1968.
- [13] P. Morrison, E. Morrison. *Charles Babbage and his calculating engines: Selected writings by Charles Babbage and others*. Dover, New York, 1961. [archive.org/details/philtrans09445034](https://archive.org/details/philtrans09445034)
- [14] P. Naur. Programming by action clusters. *BIT Numerical Mathematics*, 9(3):250–258, 1969.
- [15] P. Naur. Formalization in program development. *BIT Numerical Mathematics*, 22(4):437–453, 1982.
- [16] D.C. Oppen. Prettyprinting. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 2(4):465–483, 1980.
- [17] N. Ramsey. Literate programming simplified. *IEEE Software*, 11(5):97–105, 1994.
- [18] R.A. Snowdon. PEARL: an interactive system for the preparation and validation of structured programs. *SIGPLAN Notices*, 7(3):9–26, Mar. 1972.
- [19] E. Towster. A convention for explicit declaration of environments and top-down refinement of data. *IEEE Transactions on Software Engineering*, SE-5(4):374–386, July 1979.
- [20] T. Tuleja. *The Catalog of Lost Books: An annotated and seriously addled collection of great books that should have been written but never were*. Fawcett Columbine, 1989.
- [21] G.M. Weinberg. *The Psychology of Computer Programming*, vol. 29. Van Nostrand Reinhold New York, 1971.
- [22] M.V. Wilkes, D.J. Wheeler, S. Gill. *The Preparation of Programs for an Electronic Digital Computer: With special reference to the EDSAC and the use of a library of subroutines*. Addison-Wesley Press, 1951.

◇ Mitchell Gerrard  
mitchell dot gerrard (at) gmail dot com