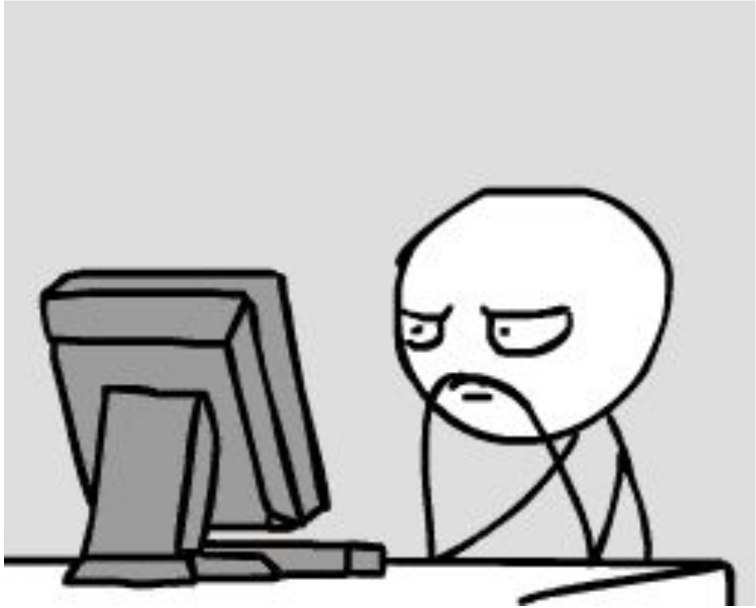


Programming and Reproducibility

Cyril Pernet, PhD

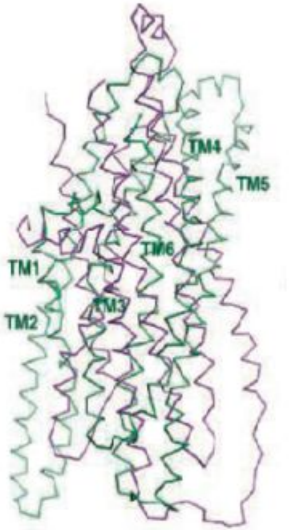
The impact of software errors

Structure of MsbA from *E. coli*: A Homolog of the Multidrug Resistance ATP Binding Cassette (ABC) Transporters

Geoffrey Chang* and Christopher B. Roth

Multidrug resistance (MDR) is a serious medical problem and presents a major challenge to the treatment of disease and the development of novel therapeutics. ABC transporters that are associated with multidrug resistance (MDR-ABC transporters) translocate hydrophobic drugs and lipids from the inner to the outer leaflet of the cell membrane. To better elucidate the structural basis for the "flip-flop" mechanism of substrate movement across the lipid bilayer, we have determined the structure of the lipid flippase MsbA from *Escherichia coli* by x-ray crystallography to a resolution of 4.5 angstroms. MsbA is organized as a homodimer with each subunit containing six transmembrane α-helices and a nucleotide-binding domain. The asymmetric distribution of charged residues lining a central chamber suggests a general mechanism for the translocation of substrate by MsbA and other MDR-ABC transporters. The structure of MsbA can serve as a model for the MDR-ABC transporters that confer multidrug resistance to cancer cells and infectious microorganisms.

www.sciencemag.org SCIENCE VOL 293 7 SEPTEMBER 2001



Geoffrey Chang

Structure of the ABC Transporter MsbA in Complex with ADP·Vanadate and Lipopolysaccharide

Christopher L. Reyes and Geoffrey Chang*

Select members of the adenosine triphosphate (ATP)–binding cassette (ABC) transporter family couple ATP binding and hydrolysis to substrate efflux and confer multidrug resistance. We have determined the x-ray structure of MsbA in complex with magnesium, adenosine diphosphate, and inorganic vanadate ($Mg·ADP·V_i$) and the rough-chemotype lipopolysaccharide, Ra LPS. The structure supports a model involving a rigid-body torque of the two transmembrane domains during ATP hydrolysis and suggests a mechanism by which the nucleotide-binding domain communicates with the transmembrane domain. We propose a lipid "flip-flop" mechanism in which the sugar groups are sequestered in the chamber while the hydrophobic tails are dragged through the lipid bilayer.

13 MAY 2005 VOL 308 SCIENCE www.sciencemag.org

X-ray Structure of the EmrE Multidrug Transporter in Complex with a Substrate

Owen Pornillos, Yen-Ju Chen, Andy P. Chen, Geoffrey Chang*

EmrE is a prototype of the Small Multidrug Resistance family of efflux transporters and actively expels positively charged hydrophobic drugs across the inner membrane of *Escherichia coli*. Here, we report the x-ray crystal structure, at 3.7 angstrom resolution, of one conformational state of the EmrE transporter in complex with a translocation substrate, tetraphenylphosphonium. Two EmrE polypeptides form a homodimeric transporter that binds substrate at the dimerization interface. The two subunits have opposite orientations in the membrane and adopt slightly different folds, forming an asymmetric antiparallel dimer. This unusual architecture likely confers unidirectionality to transport by creating an asymmetric substrate translocation pathway. On the basis of available structural data, we propose a model for the proton-dependent drug efflux mechanism of EmrE.

23 DECEMBER 2005 VOL 310 SCIENCE www.sciencemag.org

Russ Poldrack slide

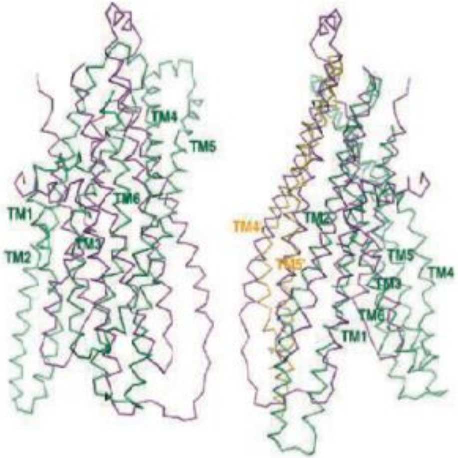
The impact of software errors

Retraction

WE WISH TO RETRACT OUR RESEARCH ARTICLE "STRUCTURE OF MsbA from *E. coli*: A homolog of the multidrug resistance ATP binding cassette (ABC) transporters" and both of our Reports "Structure of the ABC transporter MsbA in complex with ADP•vanadate and lipopolysaccharide" and "X-ray structure of the EmrE multidrug transporter in complex with a substrate" (*1*–*3*).

The recently reported structure of Sav1866 (*4*) indicated that our MsbA structures (*1*, *2*, *5*) were incorrect in both the hand of the structure and the topology. Thus, our biological interpretations based on these inverted models for MsbA are invalid.

An in-house data reduction program introduced a change in sign for anomalous differences. This program, which was not part of a conventional data processing package, converted the anomalous pairs (I+ and I–) to (F– and F+), thereby introducing a sign change. As the diffraction data collected for each set of MsbA crystals and for the EmrE crystals were processed with the same program, the structures reported in (*1*–*3*, *5*, *6*) had the wrong hand.



www.sciencemag.org SCIENCE VOL 314 22 DECEMBER 2006

Russ Poldrack slide

Computational Thinking

Thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking.

Martin Lillholm's slide

Computational Thinking

It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

LISA HANEY

Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it? Computer science rests on solid theoretical underpinnings to answer such questions precisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine's instruction set, its resource constraints, and its operating environment.

In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

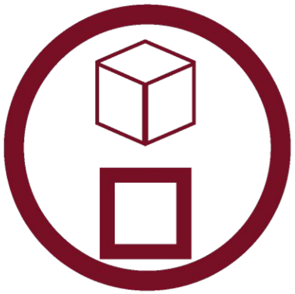
Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code. It is type checking as the generalization of dimensional analysis. It is recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name. It is recognizing both the cost and power of indirect addressing and procedure call. It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance.

Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is

Computational Thinking - How

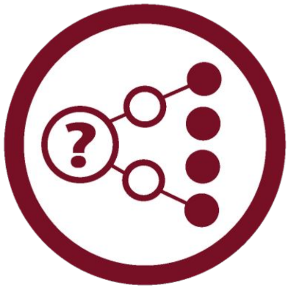
Abstraction

- Generalization
- Identify input/output
- Identify key data elements
- Ignore problem-specific details



Automation

- Design "recipe"
- Break into steps/units
- Implement recipe/algorithm
- Find the errors

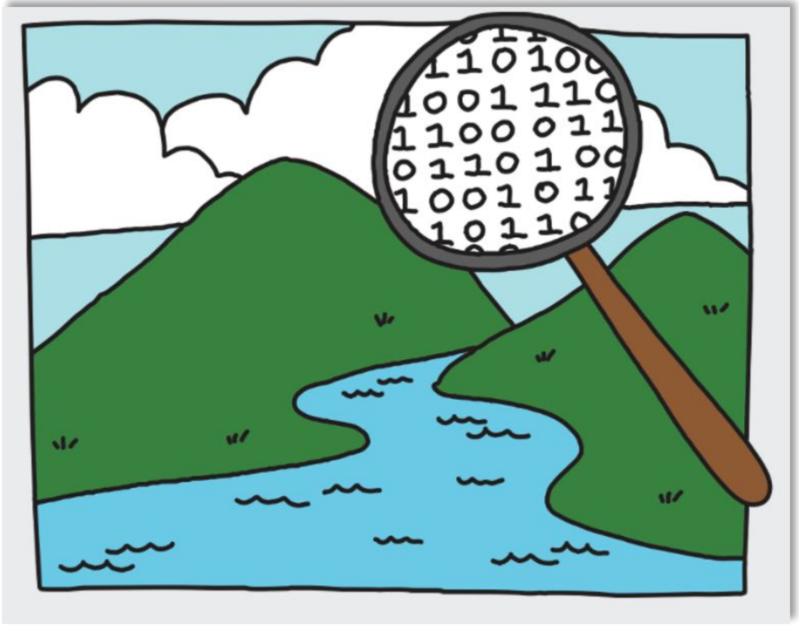


Martin Lillholm's slide

Computational Thinking - Why

Automation

- Save time
- **<u>Reusability, robustness</u>**
- **<u>Share</u>**
- Solve otherwise-impossible problems
- Objectivity
- Inspire new ideas



Martin Lillholm's slide

Algorithms

Definition

set of rules to be followed in calculations or other problem-solving operations

Anyone has an idea of how old is the 1st algorithm?

Babylonian clay tablets that dates to around 1800-1600 BC describing procedures for solving a whole class of problems, e.g. finding a square-root!

Computer algorithms

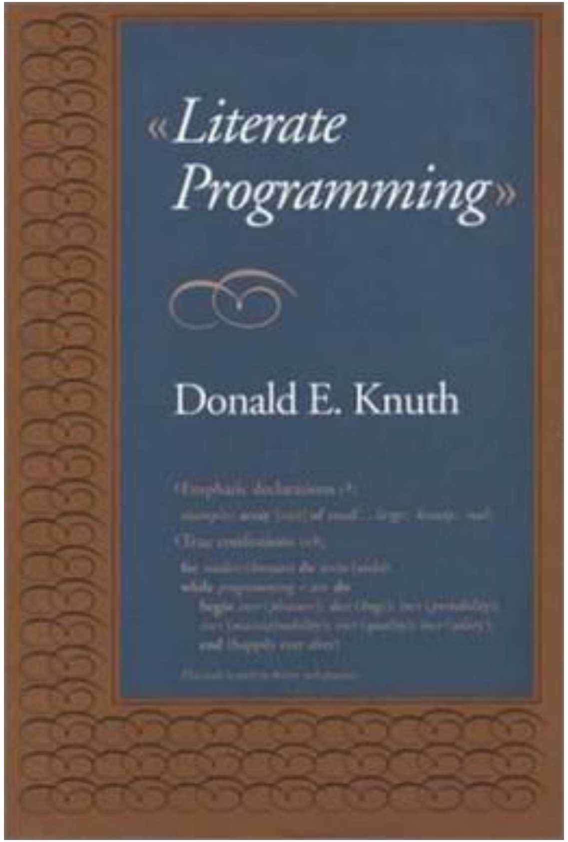
"Sketch of the Analytical Engine Invented by Charles Babbage, Esq"(Richard & John Taylor, 1843)

Ada Lovelace not only translated the work of an italian mathematician on automatic calculation machine, she added notes and equations and the algorithm to compute Bernouilli numbers

Algorithms are recipes - you must know what cake you want to make, choose the recipe that will give the expected result and get the right ingredients - but you don’t need to invent a recipe from scratch to bake a good cake.

Literate programming

What is this?



Literate programming is a programming paradigm introduced in 1984 by Donald Knuth in which a computer program is given as an explanation of how it works in a natural language, such as English, interspersed (embedded) with snippets of macros and traditional source code, from which compilable source code can be generated. The approach is used in scientific computing and in data science routinely for reproducible research and open access purposes.

Wikipedia

Literate programming vs. comments?

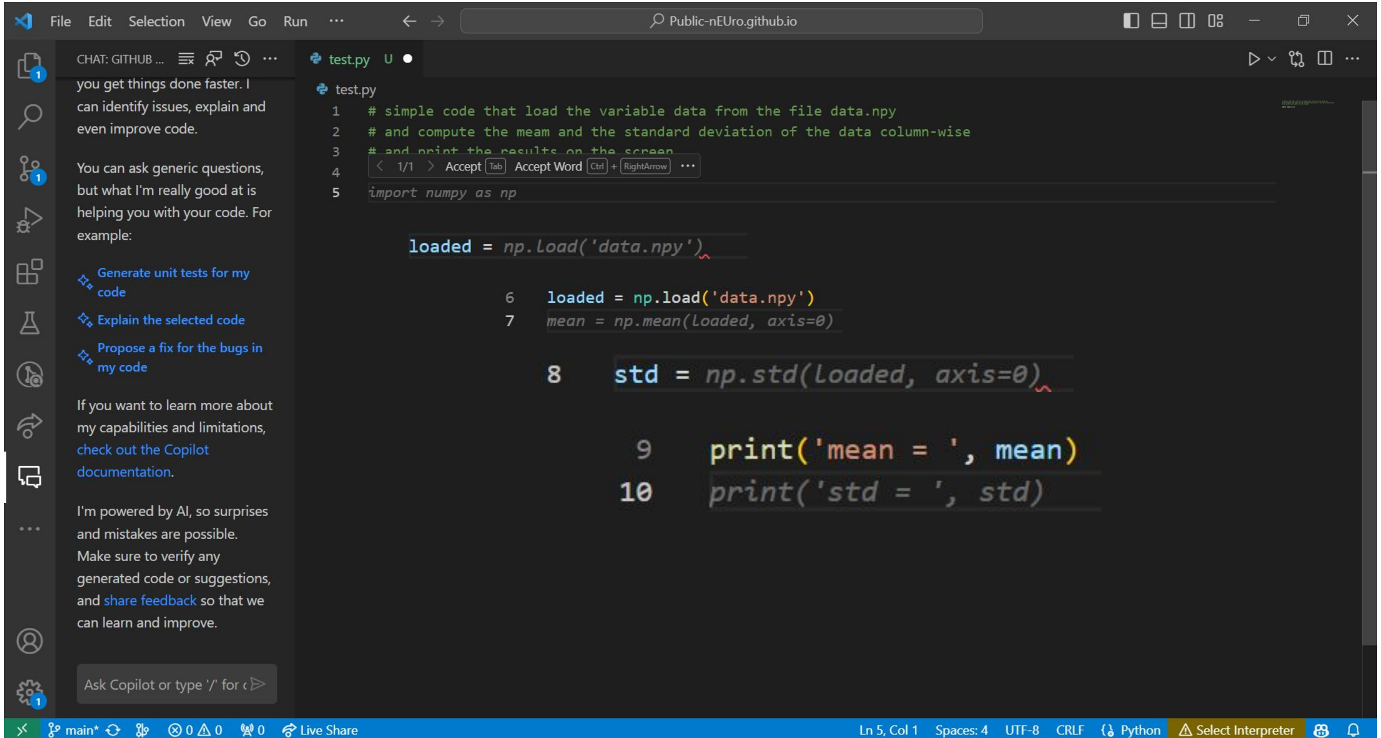
Given what we talk about in terms of computational thinking, what would be literate programming vs commenting.

```
for subject 1:20
        data.RT = load(data);
        avg(subject) = mean(data.RT)
end
result = two_samples_ttest(avg(1:10),avg(11:20))
```

Programming with AI

ChatGPT and Co-pilot

AI for coding is build from language, in particular natural language = you want literate programming



Default (GPT-3.5)

write a simple python code that load the variable data from the file data.npy
and compute the meam and the standard deviation of the data column-wise
and print the results on the screen

To accomplish this task, you can use the NumPy library in Python. You'll need to load the data from the `data.npy` file, calculate the mean and standard deviation column-wise, and then print the results to the screen. Here's a simple Python code to do that:

python   Save to grepper   Copy code

```python
import numpy as np

# Load the data from the 'data.npy' file
data = np.load('data.npy')

# Calculate the mean and standard deviation column-wise
mean_values = np.mean(data, axis=0)
std_deviation = np.std(data, axis=0)

# Print the results
print("Mean values for each column:")
print(mean_values)

print("\nStandard deviation for each column:")
print(std_deviation)
```

Make sure that the `data.npy` file is in the same directory as this Python script, or provide the full path to the file if it's located elsewhere. This code will load the data, compute the m Regenerate standard deviation for each column, and then display the results on the screen.

Coding vs. Programming vs. no code at all

Programming as Theory Building*

Peter Naur
Datalogisk Institut, Copenhagen University, Sigurdsgade 41, DK-2200 Copenhagen N, Denmark

Some views on programming, taken in a wide sense and regarded as a human activity, are presented. Accepting that programs will not only have to be designed and produced, but also modified so as to cater for changing demands, it is concluded that the proper, primary aim of programming is, not to produce programs, but to have the programmers build theories of the manner in which the problems at hand are solved by program execution. The implications of such a view of programming on matters such as program life and modification, system development methods, and the professional status of programmers, are discussed.

Keywords: General; General Terms: Human Factors, Theory, programming psychology, programming methodology.

1. Introduction

The present discussion is a contribution to the understanding of what programming is. It suggests that programming properly should be regarded as an activity by which the programmers form or achieve a certain kind of insight, a theory, of the matters at hand. This suggestion is in contrast to what appears to be a more common notion, that programming should be regarded as a production of a program and certain other texts.

- The most important Danish computer science paper (Naur, 1985)
- A program is a *theory*
- Programming is to achieve or form a theory of the matters at hand
- The program text (code) does not contain all aspects of the program (discludes all the failed previous experiments)

Coding vs. Programming

Translate a human command into a computer code		
Writing a script to analyze* data (* from description to using ML algorithms)		
Writing a script calling custom functions to analyze data		
Writing a software to analyze data		
Writing a test making sure the code does it is supposed to do		

Can you program without coding?

Do you always need to code to analyze data?

Can you program without coding?

- A program is a set of instructions, a theory about how data should be handled
- You can build a program with a user interface (e.g. Orange data mining)
- Of course there is code behind the scene, but your theory is about ‘actionables’ (re computational thinking)

> "Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent."
>
> — Cuny, Snyder, Wing, 2010

Do you always need to code to analyze data?

- no code = ease of manipulating and analyzing data with only domain expert knowledge required + a little time to learn the tool (orange, google, etc - externally validated!)
- no code = limited data analytics (proprietary softwares may not always have the wanted functionality)
- no code = no traceability / reproducibility? some tools, create the code under the GUI and let you export it