Cultures of Programming

Chapter 1

Introduction

Teacher: The author of this book believes that all have different ways of thinking about programming. I wonder if that is really the case. Maybe we can first find out whether we agree on what programming is in the first place…

Xenophon: I do not see how we could disagree on this. Programming is the process of developing a software system that solves some business problem.

Socrates: This is a painfully limited perspective! Programming is a tool for understandings the world. It can equally inspire and be used to express new creative ideas.

Archimedes: Those are lofty visions, but in reality, most programming is done to solve a business problem. I see a grain of truth in what you say though, because modern development methodologies make understanding of the problem, obtained through programming, a key part of the iterative development lifecycle.

Socrates: You keep treating programming as a boring commercial utility. It is better seen as a kind of literacy. Programming forces you to think about the world in a clear structured way

Socrates: They may be great, but they are boring. The amazing thing about computers is that they give you an unprecedented creative freedom. What you can do with a computer is more restricted by your imagination than by technical limitations!

Diogenes: The creative freedom of programming inspired the MIT hackers[5] in the 1960s who were instrumental in creating the ARPANET, a predecessor of the Internet, and early computers to choose one example that emerged from those circles and that is relevant to programming today, it would be the UNIX operating system and the C programming language.

Xenophon: I thought that C remains widely used only for historical reasons. Are they trying to find safer and more expressive alternatives these days?

Diogenes: This is a common myth, but the power of C is in that it lets you access and communicate with anything on your computer. The C language architecture captured this idea when it was created and it still follows this basic principle.[6]

Socrates: I too think that the early MIT spirit is a source of many good ideas, but I would not choose UNIX and C as the best examples. To me, these two sound like the exact opposite of systems showing the creative potential of computers. I much prefer creative use of programming like the Spacewar! game or the graphical system Sketchpad. But today, the most [illegible] instances at

Socrates: I agree that the Amazon system is unacceptable. But this is not an issue with the algorithm itself. The issue is that it was used poorly. Presumably, the algorithm just learned the bias that was already present in the training data set.[8]

Socrates: What I find more worrying is that we often do not understand the effects of programs that we create. Consider the numerous AI chatbots that have become abusive, inflammatory and racist in the language of their users or their training datasets!

Teacher: We will get to issues with programming soon enough, but I wanted to start with important achievements and positive [illegible]

Archimedes: As I said, I do not think [illegible] gradually getting better at [illegible] example is the Agile movement [illegible] developers need to work more [illegible]

Socrates: Phrases like "we value individuals [illegible] Agile manifesto sound nice. But [illegible] support people for commercial [illegible] it is a mechanism for control. [illegible] are mentioned often in leading [illegible] narrow.

Xenophon: I do not understand the [illegible] interesting, you need a large team [illegible] form of team structure or "[illegible]

Diogenes: I agree with *Socrates* that [illegible] *Xenophon* finds to be an example? The developers [illegible] ed to land a man on the moon [illegible] ed thanks to rigorous definition [illegible] ing.

Pythagoras: You are playing it safe [illegible] pressive achievement, but [illegible]

Archimedes: That is correct, but those [illegible] re the computer to avoid [illegible] mpting to fix them at the last minute.[3]

Teacher: This is why I find the present state [illegible] al entity and so you can use formal mathematical methods to show that a program is correct. We should build software [illegible] post-hoc workarounds!

Socrates: How is this usually done in practice? What is your example?

Pythagoras: Formal verification is challenging, but there are some good examples such as the formally verified microkernel seL4 that has been used as the basis for systems that are robust against, including one that controlled an unmanned flight of the AH-6 helicopter.[4] But my example of a paradigmatic achievement would be the Algol programming language, which pioneered the idea of treating programs as mathematical objects, and so enabled rigorously analysed and made all the follow-up work thinkable.

Teacher: Our examples so far include the Agile movement, the Algol language and the Apollo guidance computer. [illegible] the

Socrates: [illegible] need [illegible] build [illegible] programs [illegible]

Pythagoras: [illegible] tion [illegible] programs [illegible] be [illegible]

Diogenes: [illegible] incr [illegible]

Pythagoras: [illegible] gen [illegible] information about a specific problem in this way.

Diogenes: Even if the algorithm is general, programming still relies on implicit practical knowledge. In an artificial neural network [illegible] functions, hyperparameters and [illegible] about algorithms like neural networks is that it is very hard to gain the necessary practical experience to avoid such issues.

Pythagoras: You can build systems that guarantee fairness, but not by relying on unreliable practical experience. For systems that make decisions about individuals, you can use counterfactual fairness,[10] which ensures that the result given by the algorithm is the same regardless of the demographic group of the individual.

Xenophon: I have to admit, this discussion is beyond me. I can see that there are [illegible] problematic algorithms, but which one should I use if I need to conform with the right to explanation [illegible] that we can [illegible]

Socrates: A regular [illegible] mers needs [illegible] the discuss [illegible] cerning tec [illegible]

Teacher: We are [illegible] can use our [illegible]

Xenophon: I don [illegible] business co [illegible]

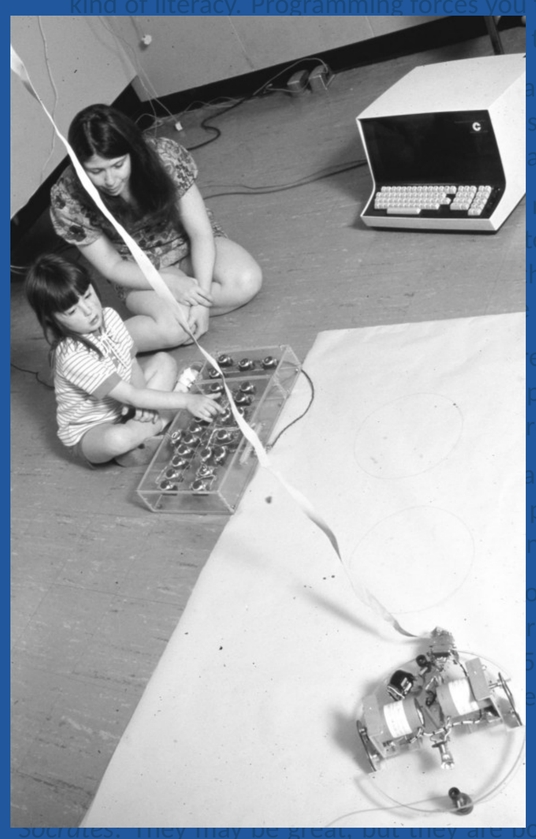
Diogenes: It will [illegible]

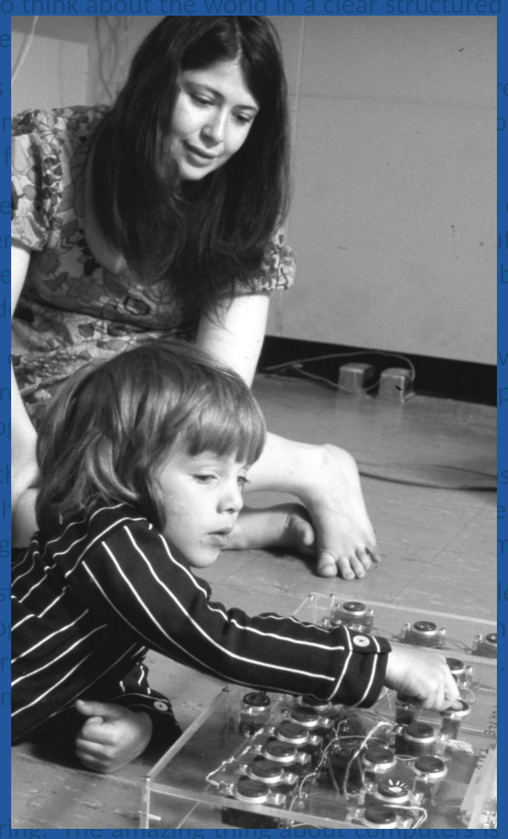
Teacher: I'm sur [illegible] tion. If we [illegible] concepts, w [illegible] the develop [illegible]

Managerial

Programmer Madeleine Carey standing next to a stack of 60,000 cards with the source code of the SAGE (Semi-Automatic Ground Environment) master program.

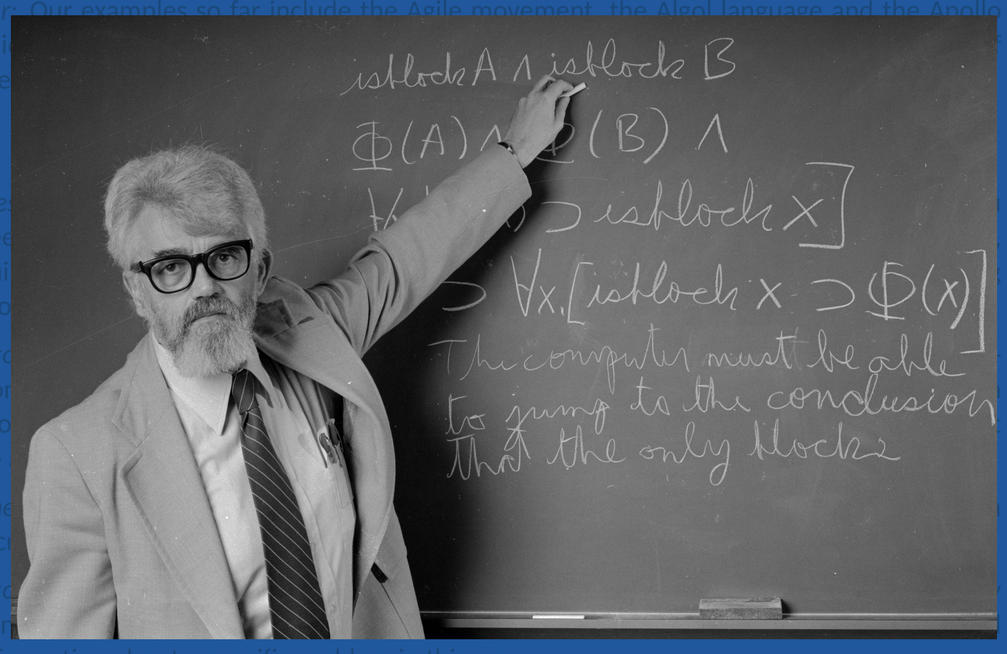
Humanistic





Radia Perlman and children with "button boxes", a simple programming device that she created for controlling "floor turtles", robots based on the Logo programming language.

Mathematical



John McCarthy, a pioneer in mathematical reasoning about computer programs, uses mathematical notation to explain an early artificial intelligence algorithm.

Engineering

A promotional photo showing an operator using the IBM System/360, a system released in 1965, a successful product whose initial engineering difficulties nearly destroyed IBM.

Hacker

Dan Edwards and Peter Samson playing Spacewar!, one of the first computer games, built in 1961 for fun and also to showcase the capabilities of the PDP-1 computer.