# “You Are Not Expected to Understand This”

## How 26 Lines of Code Changed the World

Edited by Torie Bosch

With an introduction by Ellen Ullman
and illustrations by Kelly Chudler

Princeton University Press / Princeton & Oxford

Published by Princeton University Press
41 William Street, Princeton, New Jersey 08540
99 Banbury Road, Oxford OX2 6JX

press.princeton.edu



Library of Congress Cataloging-in-Publication Data

Names: Bosch, Torie, editor. | Chudler, Kelly S., illustrator. | Ullman, Ellen, writer of introduction.
Title: You are not expected to understand this : how 26 lines of code changed the world / edited by Torie Bosch ; with an introduction by Ellen Ullman and illustrations by Kelly Chudler.
Description: First edition. | Princeton : Princeton University Press, [2022] | Includes bibliographical references and index.
Identifiers: LCCN 2022013091 (print) | LCCN 2022013092 (ebook) | ISBN 9780691208480 (pbk. ; acid-free paper) | ISBN 9780691230818 (e-book)
Subjects: LCSH: Computer programming—Popular works. | Computer science—Social aspects—Popular works. | BISAC: COMPUTERS / Programming / General | SOCIAL SCIENCE / Technology Studies
Classification: LCC QA76.6 .Y585 2022 (print) | LCC QA76.6 (ebook) | DDC 005.13—dc23/eng/20220527
LC record available at https://lccn.loc.gov/2022013091
LC ebook record available at https://lccn.loc.gov/2022013092

British Library Cataloging-in-Publication Data is available

Editorial: Hallie Stebbins, Kristen Hop, and Kiran Pandey
Production Editorial: Natalie Baan
Text and Cover Design: Chris Ferrante
Production: Danielle Amatucci and Lauren Reese
Publicity: Kate Farquhar-Thomson and Sara Henning-Stout
Copyeditor: Michele Rosen

Page 132: Comic adapted from MonkeyUser, reproduced with permission.

This book has been composed in IBM Plex

Printed on acid-free paper. ∞

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

# Contents

5

# BASIC and the Illusion of Coding Empowerment

**Joy Lisi Rankin**

During the first half of 1964, two college-age White men, John McGeachie and Michael Busch, devoted hours to computer programming. So much time, in fact, that McGeachie was known as 225, short for the GE-225 mainframe computer for which he was responsible, and Busch was known as 30, short for the GE Datanet-30 computer that he programmed. They were students at Dartmouth, an elite, overwhelmingly White, Ivy League college that admitted only men as undergraduates, and they were coding a new computing network. In the early 1960s, McGeachie's and Busch's access to technology was extraordinary.

In the 1960s, most mainframe computers ran on batch processing. Programs were communicated to the machine through inputs known as keypunch cards. Holes punched in the cards communicated numbers, letters, and symbols to the computer. One program often consisted of many cards. At the time, managers sought to keep computers running as much as possible—they were quite expensive, and organizations wanted to get their money's worth—so individual programs were grouped together and run in large groups, known as batches. For example, before Dartmouth acquired its own computer, Dartmouth professor Tom Kurtz made daytrips by train to use the MIT computer, carrying with him a box full of punched cards encoding his and his colleagues' programs: economics models, physics simulations, mathematical equations.

Typically, a computer operator handled the batch input process, as well as retrieving output such as printouts. As a result, someone who wanted to create and run a computer program had no interaction with the computer system itself—and they could wait hours or days for the results of running their program. This meant that the several thousand computers in the United States in the early 1960s were out of reach of nearly everyone, especially young people. Even the computers installed at universities were the province of a handful of faculty and graduate students. That would soon change.

The men at Dartmouth sought to challenge those limits of accessibility and batch processing. Math professor John Kemeny persuaded the trustees of the college that computing would be essential for Dartmouth students as the future leaders of American science and industry. His fellow math professor Kurtz envisioned a system where all students would be able to access computers directly, without the delays and middlemen of batch processing. Kurtz also imagined that computing would be freely available to students as part of their college experience like unfettered library access—being able to browse and pull books directly off the shelves, rather than submit a ticket for someone else to retrieve a book. Finally, Kurtz believed that Dartmouth could accomplish this by building a time-sharing network.

Time-sharing was a new form of computing in the 1960s. Time-sharing *sounds* like computer users were signing up for blocks of computing time: Alice gets 15 minutes, then Bob gets 15 minutes after Alice. But it actually means programming a mainframe computer to share its own time and computing resources among multiple programs running at the same time. In effect, this meant that multiple people could sit at individual terminals connected to one mainframe and write, run, and debug their programs at the same time.

On the Dartmouth network, the individual terminals were teletypewriter terminals that had been developed for telegraphy. They looked like old-fashioned typewriters with large printers built in. A user saw their program as they typed on the teletype, and the computer communicated results to them by printing on the teletype. Telephone wires connected teletypes to the mainframe. This meant that terminals could be—and were—located far from the network's mainframe, even in another state or halfway across the country.

In May 1964, the Dartmouth College Time-Sharing System, the early personal and social computing network that McGeachie and Busch helped program, was launched with the simultaneous and successful run of two BASIC programs. BASIC was Beginner's All-purpose Symbolic Instruction Code, a computing language developed at Dartmouth under the guiding principle that it should be easy to learn and use.

We don't know exactly what those lines of BASIC code were. We don't even know who ran the two programs.[1] But we know now that for three reasons, those BASIC programs made America's digital culture possible by spreading personal computing far, fast, and wide. The first and second reasons are fairly well known: the revolutionary accessibility of Dartmouth's computer network and the radical ease of BASIC. The third reason is the most important, yet has been overlooked: how BASIC limited paths and possibilities.

Although building a computer network for undergraduate use was visionary in the 1960s, it would not have been nearly as successful if not for BASIC. BASIC and Dartmouth's network—and the rapid uptake of both—were inseparable. Computing languages prior to BASIC, such as COBOL and FORTRAN, had been developed for scientific, research, and business purposes. They were not known for being easy to learn or user-friendly. FORTRAN's name came from FORmula TRANslation, reflecting its intended use for math and science computing.

In 1967, a student at Williams College created a program to score ski jump competitions—a challenging task that took a team of faculty and students over three hours by hand. The Williams student wrote his program in FORTRAN to run on an IBM. He spent 50 hours writing it. Meanwhile that same year, an instructor at Vermont Academy created a program to score an entire ski *meet*—ski jump plus cross-country, downhill, and slalom. The Vermont instructor wrote his program in BASIC to run on Dartmouth's network. He spent 10 hours writing it.

Compared with languages like FORTRAN or COBOL, BASIC was much faster and easier to learn. BASIC's commands—including `IF-THEN`, `LET`, `PRINT`, and `READ`—more closely resembled everyday English. At Dartmouth, the combination of BASIC and the time-sharing network enabled students to quickly write and debug short programs, to experiment, to not be afraid of making mistakes, especially because they could see the results of their programs in seconds or minutes, not days or weeks. They used BASIC for their coursework and to write letters home. They produced computer art, simulated slot machines, and programmed and played games including chess, checkers, poker, and slalom skiing. By 1968, 80 percent of Dartmouth students regularly used the network and BASIC.

In that way, BASIC offered the illusion of coding empowerment. Consider the opening of this essay: sometime in May 1964, two men sat in front of two teletypes at Dartmouth, and they successfully ran simultaneous BASIC programs on the college's brand-new time-sharing network. The fact that they were young White men at an elite, predominantly White college, is central to this story, not incidental.

During the 1960s, many women and Black people worked in computing. Before World War II, a computer was a person who performed mathematical calculations. Computers worked in business and scientific settings, and when computers became machines, many women worked *with* computers: writing programs, translating business needs to computer applications as systems analysts, operating keypunches and mainframes, and filling similar roles across industries and academic disciplines.

A 1967 issue of *Cosmopolitan* magazine with the headline "The Computer Girls" celebrated computing as "woman's work." In

*Hidden Figures*, the journalist Margot Lee Shetterly documents how she "can put names to almost 50 black women who worked as computers, mathematicians, engineers, or scientists at the Langley Memorial Aeronautical Laboratory from 1943 through 1980."[2] Likewise, the archivist Arvid Nelsen identifies at least 57 Black Americans working in computing between 1959 and 1996—just from the "Speaking of People" column in *Ebony* magazine.[3] As Claire Evans documents in her essay in this book, well-known women like Jean Sammet and Grace Hopper were not exceptions in early computing. Rather, they embodied the fact that early machine computing was a feminine field.

That shifted during the last decades of the twentieth century, when computing gained prestige in the United States and the United Kingdom by becoming the realm of affluent White men.[4] When Kemeny sold Dartmouth trustees on the idea that computing was essential knowledge for the future American leaders whom Dartmouth was producing, he was associating the power of computing with both the Whiteness and the maleness of the college. Requiring all first-year students taking math courses to successfully write a BASIC program further cemented the relationship among computing, Whiteness, affluence, and power at Dartmouth.

When other schools and universities around New England expressed interest in connecting to Dartmouth's network during the 1960s, Kemeny and Kurtz happily acquiesced. In fact, the college even secured a National Science Foundation (NSF) grant to support connecting 18 high schools around New England to the Dartmouth network. Some high-schoolers regularly woke at four in the morning to use the network.

But access to the Dartmouth network was by no means equal, and it was generally young, wealthy, White men who benefitted the most. Among the high schools connected to the Dartmouth network as part of the NSF Secondary Schools Project, the coed public schools—all predominantly White—had only 40 hours of network time each week. By contrast, the private schools—which were all male, wealthy, and almost exclusively White—had 72 hours of network time each week. In these years before the expansion of educational opportunities for American women, high school boys were still enrolling in many more math and science classes than high school girls. And it was in those math and science classes that they gained access to computing. During this decade of the Civil Rights Movement, Americans were reckoning with the myriad ways in which their public schools were separate but by no means equal. BASIC traveled in an American educational system that was already segregated by gender and race, so it ultimately amplified inequity in terms of computing access.

Kemeny and Kurtz decided to make BASIC's source code freely available so that BASIC could be (and was) implemented across many different makes and models of computers and networks. BASIC programs were stored on networks, shared in handwriting or by word of mouth, and soon circulated in books and informal newsletters, including the popular *People's Computer Company*. BASIC originated the idea that programming was something that just about anyone could do. And the echoes of that unexamined assumption perpetuate the pernicious myth today that all you need to do to succeed in tech is learn how to code.[5] BASIC made learning to code easy—but for whom?