The Women Who Invented Computer Programming

Algorithms

We get the word Algorithm from medieval Arab mathematics. According to Oxford University Press' Oxford Language website, it means "a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer."

The concept of an algorithm predates digital computers by thousands of years. For example, in the third century BCE, the Greek mathematician Eratosthenes described his ***sieve***, an algorithm to find all the prime numbers less than or equal to a given integer *n*:

1. Create a list of consecutive integers from 2 through *n*: (2, 3, 4, ..., *n*).
2. Initially, let *p* equal 2, the smallest prime number.
3. Enumerate the multiples of *p* by counting in increments of *p* from 2*p* to *n*, and mark them in the list (these will be 2*p*, 3*p*, 4*p*, ...; the *p* itself should not be marked).
4. Find the smallest number in the list greater than *p* that is not marked. If there was no such number, stop. Otherwise, let *p* now equal this new number (which is the next prime), and repeat from step 3.
5. When the algorithm terminates, the numbers remaining not marked in the list are all the primes below *n*.

A human computer with a basic grasp of mathematics can compute the prime numbers using this algorithm, ***but it requires the ability to interpret the words***.

A digital computer can't interpret the words. It offers a defined set of basic numeric operations such as addition, subtraction and comparison. We call these basic operations **machine code instructions**.

A program for a digital computer is an algorithm represented as a stream of these instructions. For example, step 1 of the sieve algorithm can be broken into simpler steps, each of which uses one instruction, something like:

```
      1 set c = 2
|→   2 put c in the list
|     3 add 1 to c
|_    4  if c < n then jump to line 2
      (continue with the rest of the algorithm)
```

To run the program, the computer would start at step 1 and obey the instructions one by one until it reaches the end.

There's an implicit assumption here that the computer has cells of memory (variables) in which it can store a number. (In the example the variable is called "c").

Line 4 is an "if test" which can interrupt the flow through the instructions and move it to another point in the list – line 2 in this case. Whether or not it does that depends on the result of the test. If the value of c is less than n, the program jumps back to line 2 and when it becomes equal to n, the program continues to the next step.

Modern High-Level programming languages such as Go and Java express these basic instructions in a more readable way, but behind the scenes, this is what's going on.

A computer program is a special case of an algorithm. It requires only mechanical mathematical skill to follow it and, crucially, no imagination.

Babbage and Lovelace

Charles Babbage defined a mechanical engine that could store numbers and obey simple instructions. He also defined the instructions – addition, subtraction, multiplication, division and calculation of a square root. The machine was programmable using instructions stored on the punched cards developed to drive Jacquard Looms. As such it was the first of its kind.

Luigi Menabrea wrote a description of Babbage's proposed machine in Italian, including a few examples of trivial programs. Ada Lovelace translated this work into English and added a more sophisticated program that demonstrated the full power of the machine. It computed one of the values of the Bernoulli sequence of numbers, which required many operations, if-tests and loops of instructions. Her algorithm was expressed in terms of the operations offered by Babbage's machine, which is why I think it was the first real computer program.

Another crucial difference between writing a program for a digital computer and writing an algorithm to be run by an intelligent human computer is that, if you make a mistake, there's a good chance that the human will spot it. A digital computer does what you tell it to do, not necessarily what you want it to do.

Ada's published program contains a mistake – it doesn't do what it's supposed to do. She knew what the result should be, so if she'd been able to run the program, she would have seen the mistake and fixed it. Unfortunately, Babbage never got his analytical engine working, so she never got the chance.

The mistake could have been a typesetting error at the printers, which she missed during her final proofreading session.

This is another difference between writing an algorithm for a human computer and expressing one in a form that a computer can make sense of: humans are not very good at the second task, and mistakes are common. The first draft of a computer program typically contains about one mistake in every ten lines of instructions. Programmers don't spend much of their day writing programs. Most of the time is spent testing them and figuring out the mistakes.

Lovelace died of cancer at 36. Babbage survived her, but he never got his computer built.

You can find a version of Ada's programming expressed as a C program here:

https://gist.github.com/sinclairtarget/ad18ac65d277e453da5f479d6ccfc20e

And a description by its author here:

https://twobithistory.org/2018/08/18/ada-lovelace-note-g.html

Early 20th Century Computing

Industry and commerce expanded, men became harder to recruit and women moved into clerical jobs, including acting as computers.

There was a huge demand for computing jobs to be done. Apart from simple accounting tasks, computers prepared tables of logarithms, mathematical functions (sines, cosines etc), actuarial tables, ready reckoners etc all had to computed, typeset and printed on paper. Commerce depended increasingly on professional computers.

Typically, the senior members of a computing team would translate a mathematician's algorithm into simple steps that the rest of the team could follow them using a desk calculators.

The military too needed computers. As artillery became more accurate, each model of a gun needed its own set of ranging tables.

Calculating machines became more sophisticated and morphed into analogue computers. They were each designed for a special purpose such as controlling a production line or ranging a gun. These computers were not programmable. Digital computers were more flexible and as they got more powerful, they displaced the analogue computers. Some lasted until the 1980s.

These videos show analogue computers in action:

A US Navy training film about using analogue computers to control naval guns: https://www.youtube.com/watch?v=s1i-dnAH9Y4

The Liverpool Tide Predicting Machine: https://www.youtube.com/watch?v=roNyTlGiz5o

In the 1930's Konrad Zuse in Berlin developed the Z1, a mechanical digital computer, but he couldn't get any funding. The Z computers and his plans for them were destroyed later in a bombing raid.

Meanwhile Cambridge don Alan Turing invented an entirely theoretical digital computer as part of a mathematical proof concerning algorithms.

The Turing Machine: https://www.youtube.com/watch?v=dNRDvLACg5Q

He was sent to the American Institute for Advance Studies IAS) where he met John Von Neumann.

Towards the end of the second World War the British, American and Russians all started working on electronic digital computers. Who got there first is moot, but Von Neumann's paper on the ENIAC was the first published description of a stored-program electronic computer so the ENIAC is often given that credit.

The ENIAC design team at Pennsylvania University was led by Eckert and Mauchly, with Von Neumann as a consultant. Von Neumann bought Turing's ideas into the design.

ENIAC held values as ten-digit decimal numbers. Its valve-based memory could hold twenty of them. (Instructions were stored in a separate plugboard.) ENIAC cost the US Department of Defense $480,000 (about $6M today.) and filled a large room.

Originally designed to produce artillery ranging tables, ENIAC was used to calculate the properties of the shaped chemical explosive charge that Von Neumann designed to trigger the first atomic bomb.

The ENIAC hardware design engineers were all men. There was an all-women team of mathematicians who wrote algorithms, and another all-woman team of research assistants who "coded" the algorithms – converted them into the machine code instructions that ENIAC could run. The coders included Von Neumann's wife Klara. (Pennsylvania Uni used the practice common at the time of hiring the wives of its researchers as research assistants.)

So, Ada Lovelace was the first programmer and, a hundred years later, the ENIAC coders were next.

Computing After the Second World War

In the early days of electronic digital computers, the equipment was very expensive. If you wanted one, you had to get a university department to design and build one for you. Companies like IBM started to make them later, initially mainly for the military. The first digital computer designed specifically for business use was Lyons Electronic Office (LEO), built for their own purposes by the British company Lyons, to automate the accounting of their national chain of cafes. They later spun off a separate division to sell computers and computing consultancy.

NASA was an early user of electronic computers, although in the 1960's they still employed teams of human computers. The film "Hidden Figures" tells the story of a team of African-American women computers who taught themselves to program. According to the film, they were the only ones who could figure out how NASA's shiny new computer worked. (Feasible, since it just automated what they did all day.)

With the invention of transisters and then integrated circuits, Moore's law kicked in and computers became steadily more powerful and cheaper. (Moore's law predicts that computing capacity doubles every eighteen months - you get twice the computing power for the same money or the same power for half the price.) Still, until the 1970s they remained in the province of governments and large companies. At first, graduate mathematicians, mainly men, wrote the algorithms, and coders, mainly women, turned them into runnable programs. Smaller companies continued to use rooms full of human computers cranking desk calculators.

By the seventies, mini computers (the size of a large fridge) became cheap enough for medium-sized companies to own them, which increased the demand for programmers. Later, the introduction of desktop microcomputers caused that job market to explode.

From the 1960s onwards, high level languages emerged, the earliest successful one being COBOL, mainly credited to Grace Hopper (later Rear Admiral) of the US Navy. This made writing programs much easier.

COBOL was easy to read but very wordy. Later languages were more terse. In the C programming language you can express the four lines of basic instructions shown earlier as:

```
For (c=2; c <=n; c++) {

    List[c] = c

}
```

Which is easier to get right **and** looks much more impressive.

At the same time the expanding IT industry increased the status and salaries of programmers, and drew more male graduates into the field.

Somehow, managers of programming teams (who were generally not themselves programmers) seemed to get the idea that programming in high level languages was much too hard for mere women (although it was actually much easier than working in machine code). Also, girls at school got the idea that programing wasn’t for them.

By the time I was taught Computer Science to undergraduates in the 1980s, about one in twenty of our students were women, and most of them went on to do IT jobs other than programming.