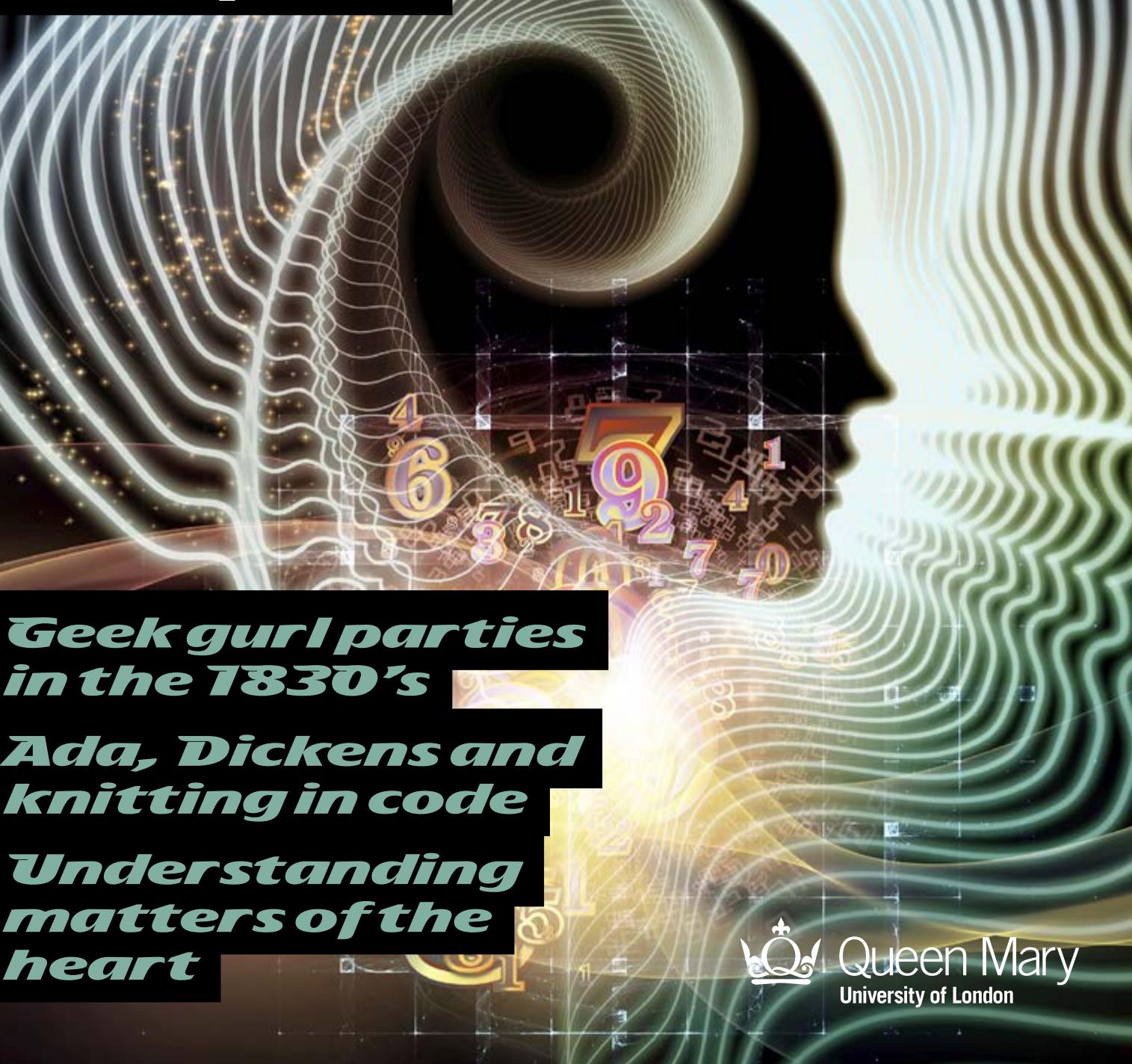


CS4FN

Computer Science for Fun

Issue 20

Ada Lovelace: the computer scientist without a computer



***Geekgurl parties
in the 1830's***

***Ada, Dickens and
knitting in code***

***Understanding
matters of the
heart***



Queen Mary
University of London

Ada Lovelace

2015 is the 200th anniversary of Ada Lovelace's birth. Famous as 'the first programmer' her vision of computer science was far wider. To celebrate, this issue explores her life, her ideas and where modern research has taken some of those ideas. We look at how women's research is still at the forefront of interdisciplinary computer science. We will also see how her work linked to the very modern idea of computational thinking.



Image of Ada as a child with the kind permission of the Principal and Fellows of Somerville College Oxford

Geek gurl parties in the 1830's

by Jane Waite, QMUL

In the 21st century, you might subscribe to a YouTube channel like the Geek Gurl Diaries, and watch videos to learn how to program a BBC micro:bit, but in 1830 where could you hang out and talk tech? Swanky parties that's where!

Sophia De Morgan, the wife of one of Ada's tutor's, wrote of Ada when she saw one of Babbage's machines:

"young as she was, she understood its working, and saw the great beauty of the invention".

In the mid 1800's, if you moved in the right circles you might get an invite to one of Charles Babbage's soirees, fancy food, dancing, a duke or two and the Difference Engine. Babbage invented the first automatic calculator, a number cruncher that used cogs, wheels and no electricity. If you were lucky he might have shown you his shiny brass test model that whirred and pinged, or reveal his complex hand written design for his newest invention, an Analytical Engine, a mechanical computer.

Ada Lovelace, met Babbage at one such party and talked tech. She had studied maths rather than the typical girl subjects of the time as her mother was worried she would turn out to be wild, like her esteemed father Lord Byron. Yes, the 'mad, bad and dangerous to know' Romantic Poet!

Many of the party goers would not have had a clue about Babbage's' new fangled contraptions, but Ada did. At only 18 she got on the program... so to speak... becoming a Victorian 'research assistant' helping on several of Babbage's projects. She translated engineering articles from French, added her own notes, and published an algorithm to work out a sequence of numbers called the Bernoulli numbers for the Analytical Engine. Some say she was the first ever programmer.

Ada saw that the Analytical Engine was more than Babbage had intended, it was not just for maths. She suggested that it might create music and wrote that the Engine 'weaves algebraic patterns just as the Jacquard loom weaves flowers and leaves.'

Ada, sure was ahead of her time: a Victorian Geek Gurl.

Why not hang out online with modern day Geek Gurl, Carrie Anne Philbin and see if you can make your own tiny Pi computer with her help? Just search for Geek Gurl Diaries.



The Silver Lady

by Jane Waite, QMUL

Babbage built machines and he bought them too. He is said to have owned the Silver Lady automata: a mechanical figure. She was an elegant Victorian lady that bowed and moved an eyeglass before then delicately peering through it. When inviting Ada to one of his parties, Babbage wrote "I hope you intend to patronize the 'Silver Lady'. She is to appear in new dresses and decorations." We don't know what Ada thought of this mechanical woman but these clockwork figures were popular parlour entertainment of the time. No TV, iPads or Xboxes to keep you amused. Robot toys are back in fashion though!

Some of these mimicking machines still exist such as the three French automata of the musician, the draughtsman and the writer, created in the 18th century by Pierre Jaquet Droz. These clockwork toys did just what their titles imply, but in a very simple way.

Perhaps Ada saw such automata and compared them to the prototype pieces of the machines she was working with: Charles Babbage's number crunching Difference Engine and the complex designs of the computing Analytical Engine. Perhaps she saw the similarities as well as the differences. She certainly saw beyond Babbage's view of his inventions, concluding that something like the Analytical Engine might one day be used to compose music. This ability to see differences and similarities and draw conclusions about how something can be used in wider ways is a key skill of the computer scientist. A fundamental part of the computational thinking skill set, it is called generalisation. Computer scientists use it to magpie ideas: collecting them, then reusing and repurposing them in new and novel ways.

Ada wrote of the Analytical Engine
"(it) might act upon other things besides numbers."



Dickens Knitting in Code

by Paul Curzon, QMUL

Charles Dickens is famous for his novels highlighting Victorian social injustice. Despite what people say, art and science really do mix, and Dickens certainly knew some computer science. In his classic novel about the French Revolution, A Tale of Two Cities, one of his characters relies on some computer science based knitting.

Dickens actually moved in the same social circles as Charles Babbage, the Victorian inventor of the first computer (which he designed but unfortunately never managed to build) and Ada Lovelace the mathematician programmer of those first computers. They went to the same dinner parties and Dickens will have seen Babbage demonstrate his prototype machines. An engineer in Dickens' novel, Little Dorrit, is even believed to be partly based on Babbage. Dickens was probably the last non-family member to visit Ada before she died. She asked him to read to her, choosing a passage from his book Dombey and Son in which the son, Paul Dombey, dies. Like Ada, Paul Dombey had suffered from illness all his life.

So Charles Dickens had lots of opportunity to learn about algorithms! His novel 'A Tale of Two Cities' is all about the French Revolution, but lurking in the shadows is some computer science. One of the characters, a revolutionary called Madame Defarge takes the responsibility of keeping a register of all those people who are to be executed once the revolution comes to pass: the aristocrats and "enemies of the people". Of course in the actual French Revolution lots of aristocrats were guillotined precisely for being enemies of the new state.

Now Madame Defarge could have just tried to memorize the names on her 'register' as she supposedly had a great memory, but the revolutionaries wanted a physical record. That raises the problem, though, of how to keep it secret, and that is where the computer science comes in. Madame Defarge knits all the time and so she decides to store the names in her knitting.

"Knitted, in her own stitches and her own symbols, it will always be as plain to her as the sun. Confide in Madame Defarge. It would be easier for the weakest poltroon that lives, to erase himself from existence, than to erase one letter of his name or crimes from the knitted register of Madame Defarge!"

Computer scientists call this Steganography: hiding information or messages in plain sight, so that no one suspects they are there at all. Modern forms of steganography include hiding messages in the digital representation of pictures and in the silences of a Skype conversation.

Madame Defarge didn't of course just knit French words in the pattern like a victorian scarf version of a T-shirt message. It wouldn't have been very secret if anyone looking at the resulting scarf could read the names. So how to do it? In fact, knitting has been used as a form of steganography for real. One way was for a person to take a ball of wool and mark messages down it in Morse Code dots and dashes. The wool was then knitted into a jumper or scarf. The message is hidden! To read it you unpick it all and read the morse code back off the wool.

That wouldn't have worked for Madame Defarge though. She wanted to add the names to the register in plain view of the person as they watched and without them knowing what she was doing. She therefore needed the knitting patterns themselves to hold the code. It was possible because she was both a fast knitter and sat knitting constantly so it raised no suspicion. The names were therefore, as Dickens writes: "Knitted, in her own stitches and her own symbols".

"Computer scientists call this Steganography: hiding information or messages in plain sight"



She used a ‘cipher’ and that brings in another area of computer science: encryption. A cipher is just an algorithm – a set of rules to follow – that converts symbols in one alphabet (letters) into different symbols. In Madame Defarge’s case the new symbols were not written but knitted sequences of stitches. Only if you know the algorithm, and a secret ‘key’ that was used in the encryption, can you convert the knitted sequences back into the original message.

In fact both steganography and encryption date back thousands of years (computer science predates computers!), though Charles Dickens may have been the first to use knitting to do it in a novel. The Ancient Greeks used steganography. In the most famous case a message was written on a slave’s shaved head. They then let the hair grow back. The Romans knew about cryptographic algorithms too and one of the most famous ciphers is called the Caesar cipher as Julius Caesar used it when writing letters...even in Roman times people were worried about the spies reading their equivalent of emails.

Dickens didn’t actually describe the code that Madame Defarge was using so we can only guess...but why not see that as an opportunity and (if you can knit) why not invent a way yourself. If you can’t knit then learn to knit first and then invent one! Somehow you need a series of stitches to represent each letter of the alphabet. In doing so you are doing algorithmic thinking with knitting. You are knitting your way to being a computer scientist.

In the Second World War, the United States censors held on to a letter that contained a knitting pattern so they could knit the jumper in case it did contain a message. Ultimately they banned people from posting knitting patterns overseas at all (along with playing chess by post) in case people were hiding messages in them.

Charles Babbage worked on codes too, and just as happened to the team at Bletchley Park in WWII, his work in the Crimean War was kept a military secret leading to others gaining the credit.

Ada Lovelace in her own words

by Ursula Martin, University of Oxford

Charles Babbage invented wonderful computing machines. But he was not very good at explaining things. That's where Ada Lovelace came in. She is famous for writing a paper in 1843 explaining how Charles Babbage's Analytical Engine worked – including a big table of formulas which is often described as "the first computer program".

Charles Babbage invented his mechanical computers to save everyone from the hard work of doing big mathematical calculations by hand. He only managed to build a few tiny working models of his first machine, his Difference Engine. It was finally built to Babbage's designs in the 1990s and you can see it in the London Science Museum. It has 8,000 mechanical parts, and is the size of small car, but when the operator turns the big handle on the side it works perfectly, and prints out correct answers.

Babbage invented, but never built, a more ambitious machine, his Analytical Engine. In modern language, this was a general purpose computer, so it could have calculated anything a modern computer can – just a lot more slowly. It was entirely mechanical, but it had all the elements we recognize today – like memory, CPU, and loops.

Lovelace's paper explains all the geeky details of how numbers are moved from memory to the CPU and back, and the way the machine would be programmed using punched cards.

But she doesn't stop there – in quaint Victorian language she tells us about the challenges familiar to every programmer today! She understands how complicated programming is:

"There are frequently several distinct sets of effects going on simultaneously; all in a manner independent of each other, and yet to a greater or less degree exercising a mutual influence."

the difficulty of getting things right:

"To adjust each to every other, and indeed even to perceive and trace them out with perfect correctness and success, entails difficulties whose nature partakes to a certain extent of those involved in every question where conditions are very numerous and inter-complicated."

and the challenge of making things go faster:

"One essential object is to choose that arrangement which shall tend to reduce to a minimum the time necessary for completing the calculation."

She explains how computing is about patterns:

"it weaves algebraical patterns just as the Jacquard-loom weaves flowers and leaves".

and inventing new ideas

"We might even invent laws ... in an arbitrary manner, and set the engine to work upon them, and thus deduce numerical results which we might not otherwise have thought of obtaining".

and being creative. If we knew the laws for composing music:

"the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent."

Alan Turing famously asked if a machine can think – Ada Lovelace got there first:

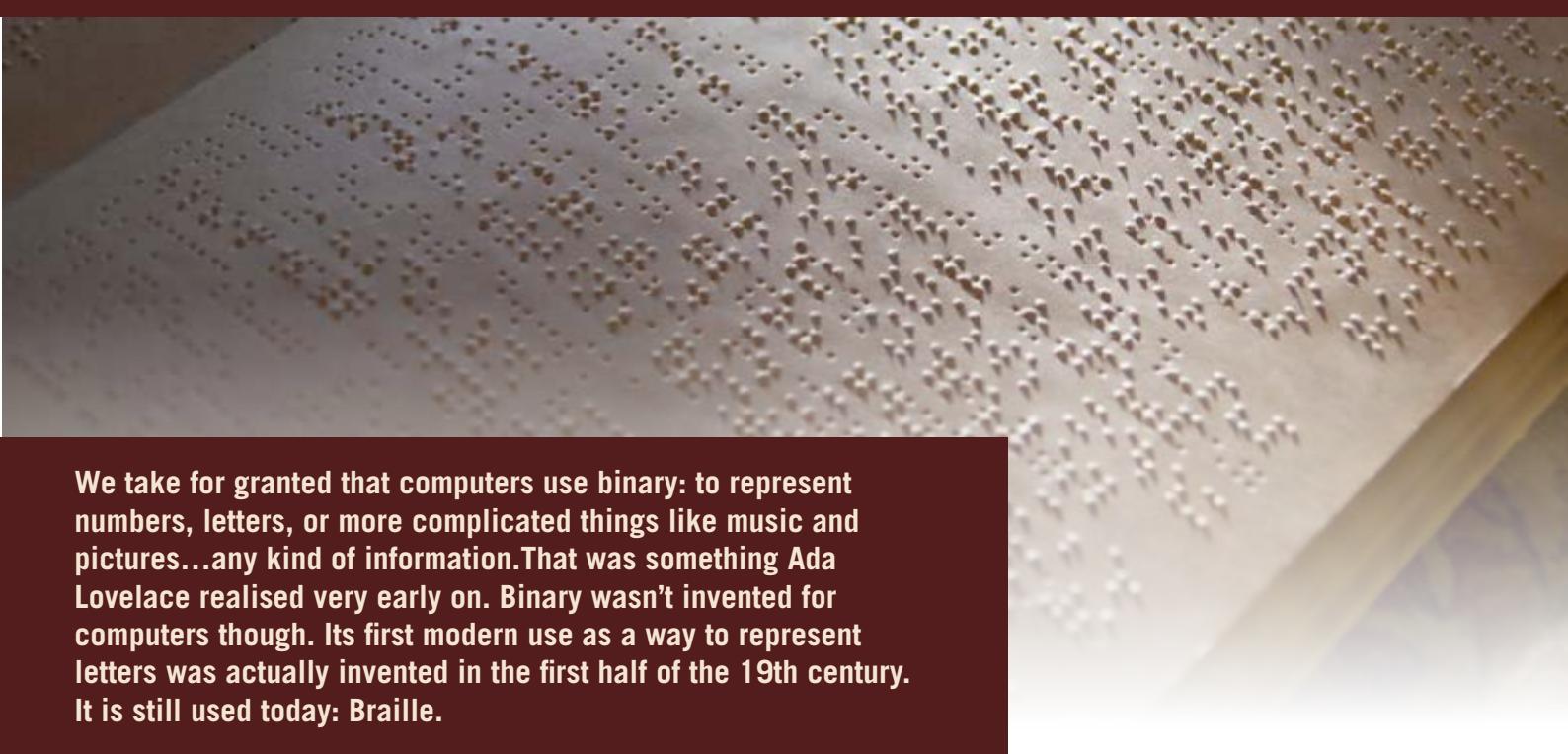
"The Analytical Engine has no pretensions whatever to originate anything. It can do whatever we know how to order it to perform."

Wow, pretty amazing, for someone born 200 years ago.

You can read the whole of Lovelace's famous paper, and play with a simulation of the Analytical Engine, at www.fourmilab.ch/babbage/contents.html

Letters from the Victorian Smog

by Paul Curzon, QMUL



We take for granted that computers use binary: to represent numbers, letters, or more complicated things like music and pictures...any kind of information. That was something Ada Lovelace realised very early on. Binary wasn't invented for computers though. Its first modern use as a way to represent letters was actually invented in the first half of the 19th century. It is still used today: Braille.

Braille is named after its inventor, Louis Braille. He was born 6 years before Ada, though they probably never met as he lived in France. He was blinded as a child in an accident and invented the first version of Braille when he was only 15 in 1824 as a way for blind people to read. What he came up with was a representation for letters that a blind person could read by touch.

Choosing a representation for the job is one of the most important parts of computational thinking. It really just means deciding how information is going to be recorded. Binary gives ways of representing any kind of information that is easy for computers to process. The idea is just that you create codes to represent things made up of only two different characters: 1 and 0. For example, you might decide that the binary for the letter 'p' was: 01110000. For the letter 'c' on the other hand you might use the code, 01100011. The capital letters, 'P' and 'C' would have completely different codes again. This is a good representation for computers to use as the 1's and 0's can themselves be represented by high and low voltages in electrical circuits, or switches being on or off.

The first representation Louis Braille chose wasn't great though. It had dots, dashes and blanks - a three symbol code rather than the two of binary. It was hard to tell the difference between the dots and dashes by touch, so in 1837 he changed the representation – switching to a code of dots and blanks. He had invented the first modern form of writing based on binary.

Braille works in the same way as modern binary representations for letters. It uses collections of raised dots (1s) and no dots (0s) to represent them. Each gives a bit of information in computer science terms. To make the bits easier to touch they're grouped into pairs. To represent all the letters of the alphabet (and more) you just need 3 pairs as that gives 64 distinct patterns. Modern Braille actually has an extra row of dots giving 256 dot/no dot combinations in the 8 positions so that many other special characters can be represented. Representing characters using 8 bits in this way is exactly the equivalent of the computer byte.

Modern computers use a standardised code, called Unicode. It gives an agreed code for referring to the characters in pretty well every language ever invented including Klingon! There is also a Unicode representation for Braille using a different code to Braille itself. It is used to allow letters to be displayed as Braille on computers! Because all computers using Unicode agree on the representations of all the different alphabets, characters and symbols they use, they can more easily work together. Agreeing the code means that it is easy to move data from one program to another.

The 1830s were an exciting time to be a computer scientist! This was around the time Charles Babbage met Ada Lovelace and they started to work together on the Analytical Engine. The ideas that formed the foundation of computer science must have been in the air, or at least in the Victorian smog.



Vulgar coloured pens

by Jane Waite, QMUL

Ada tutored students in maths using what seemed to be rather unorthodox means at the time. She suggested they used coloured pens, for example to make diagrams clearer, instead of everything being black and white, which was thought of as ‘vulgar’ at the time. Using colour to help us better understand a concept and make software easier to use is now common in the design of user interfaces.

Colour Theory looks at how designs can be improved through various aspects of colour such as complementation, contrast and vibrancy. Simply put complementation is how colours balance, contrast is how colours differ and vibrancy is how colours make us feel. User Interface designers spend time thinking about things like how to direct attention to the right place, focus the eye without straining it, and how to create an experience that is appropriate to the context. They might decide to use, for example, a bright red warning sign in stark narrow

Charles Babbage also understood the importance of colour when presenting results, if obsessively. When planning how his machine the Difference Engine would print results, he ran an exhaustive experiment. He printed the same information over and over using every combination of ink and paper colour he could get hold of to see which would be best for the eye. He put the resulting sheets into 21 books of examples. It really was obsessive: it even included black on black. Computer scientists tend to think every last detail matters.

letters, simple sweeping dark lettering on a light complementary background for a poem. The screen might turn red to attract attention if you’ve made a mistake entering data, only returning to green when the data is valid. Different colours might be used to show that you are in an editing mode rather than viewing mode of a document editor. A website might use colour coding of the pages to show the different grouped areas a visitor is viewing.

Ada seems to have been a natural as a user interface designer as well as a programmer and computational thinking wizard.

**Tut, tut!
Look
at your
labels
dear!**

by Jane Waite, QMUL



As well as showing her students how to use colour, Ada Lovelace also demanded good naming standards! She apparently scolded her students if the labels of their diagrams were not distinct, i.e. sensible and unique.

Again computer scientists like this too. When objects or variables are named in programs or the description of algorithms, developers make sure the names are unique and easy to understand, so they do not get muddled up. In very large projects, there might be entire departments whose job it is to make sure names of things are ‘just right’ with a standardised format, memorable wording and distinct terms. This job of looking after data can be called a data analyst or data architect, so Ada could do that too!

Imagine a diagram of the London Underground with each station given a nonsensical name such as azev6, Peter, 8710 with no colour coding and all the lines with the same name: ‘Line’. You might get the idea travelling on the Moscow Metro! Station names are just strings of random squiggles if you don’t speak Russian and don’t know the alphabet. Trying to spot if each station name matches the squiggles of your destination at each stop is really, really hard. Speakers of other languages that use different alphabets might have the same problem in London.

Puzzling Victorian Doodles

by Paul Curzon, QMUL

Buried in the archive of Ada's personal papers in the Bodleian Library at Oxford is a piece of paper that at first sight just looks like a bunch of Doodles with odd scribbles in Ada Lovelace and Charles Babbage's hand. They had been trying to solve a, by then, century-old puzzle that laid the foundation for a core way that computer scientists now represent data.

Their doodles include a simple map that any computer scientist would recognize – it shows the bridges of Königsberg in Prussia. Other doodles show variations with a different set of bridges. The town was split in two by the river Pregel, with two large islands in the middle. The islands were connected to the two sides of the river and each other by a series of 7 bridges. Back in the 1700's it was used as the basis of a puzzle. Could you come up with a walking tour of the whole city that would involve crossing every bridge once, but only once?

The puzzle was solved by mathematician Leonhard Euler back in 1736, but it was not so much his solution but his way of going about solving it that was revolutionary. He used what are now core parts of a computer scientist's computational thinking toolkit.

What Euler realised was that the puzzle was easier to solve if you threw away most of the information on the map of the town. That is just the computational thinking idea of

abstraction: problems are easier to solve if you hide all unnecessary information. All you need for the bridge problem is information showing the different land masses and how they are connected (i.e. the bridges between them). He made it really simple and drew small circles for each land mass (a circle for each of the two islands and the two sides of the river). He then drew lines between each pair of circles that were connected by bridges. A computer scientist calls a picture, or 'representation', like this a graph. The lines are called the 'edges' and the circles the 'nodes' of the graph. The problem now becomes a question of whether you can visit every node in the graph following each edge once and only once. The simplified picture given by the graph allowed Euler to see, when combined with some logical thinking, the answer to the problem.

These ideas became the foundation of what is now called graph theory. Just as they helped Euler see a simpler solution to the puzzle, graphs now help computer

scientists solve all sorts of problems and are the basis of all sorts of computing, from the way satnavs find the best routes, to how computers organise data to make it easier to search. It's not surprising that Ada and Charles found the puzzle interesting. Surprisingly, whilst there are lots of doodles that look like graphs, none match Euler's graph of the puzzle. The drawings show routes that solve the variations of the puzzle they invented, presumably as they explored the problem. A scribbled note also suggests they did work out the solution to the actual puzzle too.

Rather than give you the solution, draw the graph and try to solve it yourself. The solution is linked from this article at www.cs4fn.org/ada/ where you can also find out more about the ways computer scientists use graphs to solve problems.

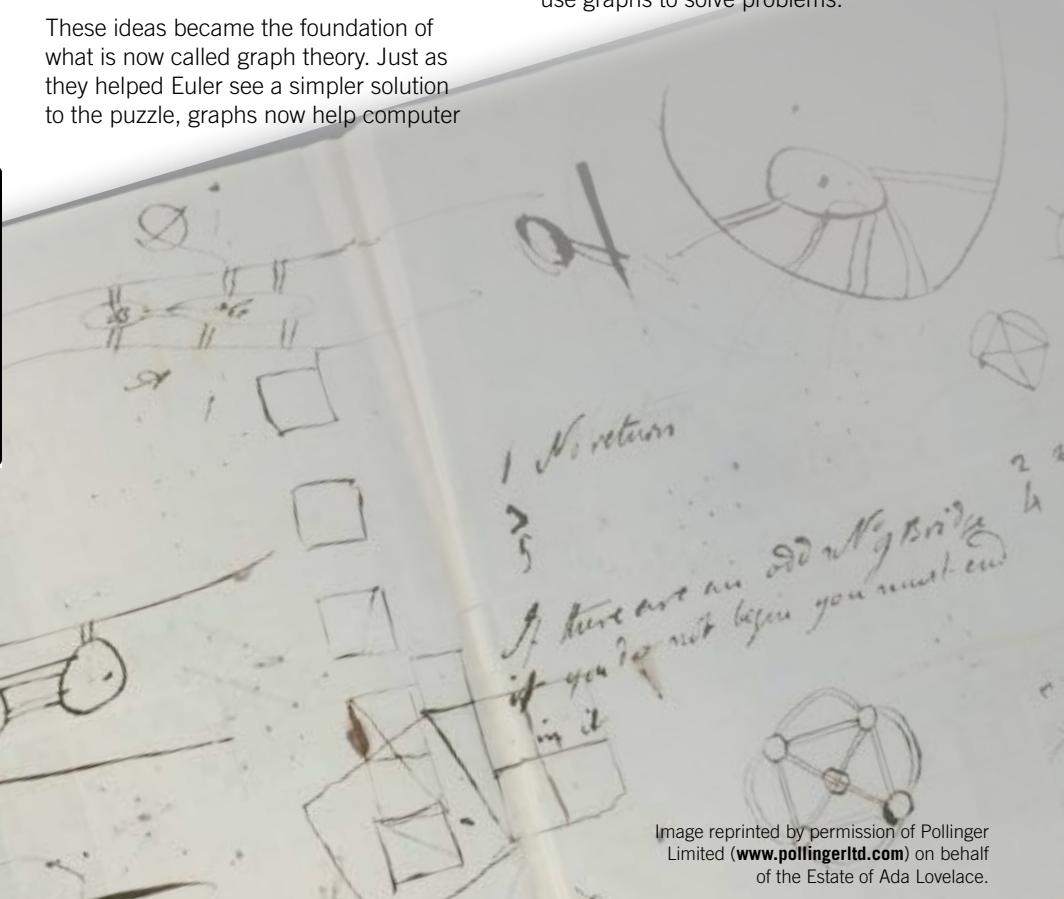
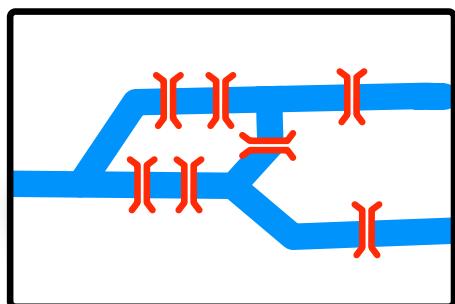


Image reprinted by permission of Pollinger Limited (www.pollingerltd.com) on behalf of the Estate of Ada Lovelace.

Frankenstein's Monster

by Paul Curzon, QMUL

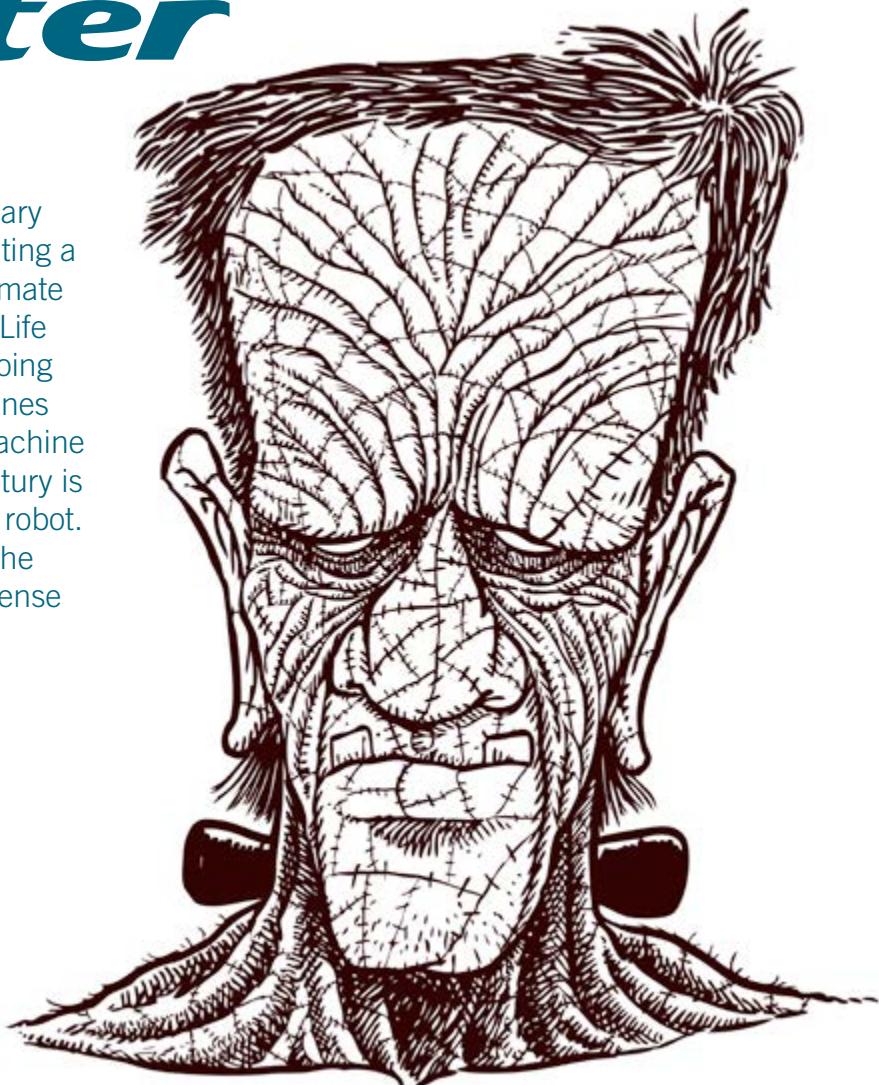
Shortly after Ada Lovelace was born, Mary Shelley, a friend of her father's, was writing a novel. In her book, Frankenstein, inanimate flesh is brought to life using electricity. Life it may not be, but engineers are now doing pretty well in creating humanoid machines that can do their own thing. Could a machine ever be considered alive? The 21st century is undoubtedly going to be the age of the robot. Maybe it's time to start thinking about the consequences in case they do gain a sense of self.

Pass the Screwdriver Please, Igor

Frankenstein was a scientist obsessed with creating life. In Shelley's original story, he succeeded, though his creation was treated as a 'Monster' struggling to cope with the gift of life it was given. The film Blade Runner explored similar ideas about how intelligent life is created, in this case androids that believe they are human, and the consequences for the creatures concerned.

Fiction? Not totally. Several groups of computer scientists are exploring what it means to create non-biological life, and how it might be done. Some are looking at robot life, working at the level of insect life-forms, for example. Others are looking at creating intelligent life within cyberspace.

For 60 years or more scientists have tried to create artificial intelligences. They have had a great deal of success in specific areas such as computer vision and chess playing programs. However none of these programs really cuts it as being life or even intelligent in the way humans are. A small band of computer scientists have been trying a different approach that they believe will ultimately lead to the creation of new life forms. Life forms that could even claim to be conscious (and who would we be to disagree with them if they do?) These scientists believe life can't be engineered in a piecemeal way, but that the whole being has to be created as



a coherent whole. Their approach is to build the basic building blocks and let life emerge from them.

Emerging Life

The general idea can be seen in part in Sodarace (www.sodarace.net), where you can build your own creatures that move around a virtual world. One approach to building creatures, such as a spider, would be to try and work out mathematical equations about how each leg moves and program those equations. The alternative, artificial life, way is to instead program the laws of physics such as gravity and friction and how masses, springs and muscles behave according to those laws. Then just put these basic bits together in a way that corresponds to a spider. With this approach you don't have to work out in advance every eventuality (What if it comes to a wall? What about a pit?) and write code to deal with it. Instead

natural behaviour emerges. The artificial life community believe, not just life-like movement, but life-like intelligence, and maybe life itself, can emerge in a similar way. Rather than programming the behaviour of muscles you program the behaviour of neurons and then build brains out of them, together with the basic biochemistry of an immune system and the like.

Want to know more? – then read Steve Grand's book: Creation, on how he created what has been claimed to be "the nearest thing to artificial life yet"...and started life as the game "Creatures". Then have a go at creating an artificial life yourself.

A Storm in a Bell Jar

by Paul Curzon, QMUL

Ada Lovelace was close friends with John Crosse, and knew his father Andrew: the ‘real Frankenstein’. Andrew Crosse apparently created insect life from electricity, stone and water...

Andrew Crosse was a ‘gentleman scientist’ doing science for his own amusement including work improving giant versions of the first batteries called ‘voltaic piles’. He was given the nickname ‘the thunder and lightning man’ because of the way he used the batteries to do giant discharges of electricity with bangs as loud as cannons. He hit the headlines when he appeared to create life from electricity, Frankenstein-like. This was an unexpected result of his experiments using electricity to make crystals. He was passing a current through water containing dissolved limestone over a period of weeks. In one experiment, about a month in, a perfect insect appeared apparently from no-where, and soon after started to move. More and more

insects then appeared. He mentioned it to friends, which led to a story in a local paper. It was then picked up nationally. Some of the stories said he had created the insects, and this led to outrage and death threats over his apparent blasphemy of trying to take the position of God. (Does this start to sound like a modern social networking storm, trolls and all?) In fact he appears to have believed, and others agreed, that the mineral samples he was using must have been contaminated with tiny insect eggs, that just naturally hatched. Scientific results are only accepted if they can be replicated. Others, who took care to avoid contamination couldn’t get the same result. The secret of creating life had not been found.

Sadly perhaps, for the story’s sake, while Shelley did know Crosse, he can’t have been the inspiration for Frankenstein as has been suggested, given she wrote the book decades earlier!

Following Faraday

by Jane Waite, QMUL

Ada was interested in all things scientific and was certainly interested in electricity. As well as working with Babbage she conversed by letter, not email, text or twitter, with Michael Faraday. Faraday was a very important scientist of the day, the head of the Royal Institution, making significant discoveries in physics and chemistry. His work on electromagnetism laid foundations for electricity to become the everyday essential we now can’t do without. She wrote:

“Perhaps no one has ready your paper with such full appreciation as myself of its practical bearings; or has valued it so justly, both for its contents, & as presented to me by its Author, for whom I entertain an esteem little short of reverence!”

Ada sure knew who was who in scientific circles. She presumably would have ‘liked’ his posts, followed him on Twitter and been a friend on Facebook!

Brain matter

by Jane Waite, QMUL



Ada Lovelace wasn't just interested in what controlled machines, she was also interested in what controlled her behaviour and that of others. Perhaps this was because her father, Lord Byron, was so wild and she was worried she might develop the same personality traits. Certainly her mother saw mathematics as a means to calm Ada and divert her from the possible excesses of poetry!

Ada studied a popular 19th century scientific subject called phrenology which asserted that the brain was made up of different parts, 'organs', each organ controlling different aspects of our character. The 'firmness organ' was sited right at the top middle of our heads, the 'hope organ' below that and forward a little towards the nose, with the rather unpleasant 'destructiveness organ' occupying the area just behind the ears. So they believed the structure of a person's brain influenced a person's behaviour.

Phrenology charts were drawn of a person's skull showing the sizes of their 'organs' based on an assessment of the person's personality. The more hopeful you were, the bigger your 'hope organ', the more musical a person the larger their 'tune organ'. Phrenology was taken very seriously by the scientific community during the early 1800's, though science has since shown there is nothing to it.

Ada was also very interested in mesmerism, a form of hypnosis but with a twist. The twist was that who ever mesmerised you, transferred their 'animal magnetism' through 'ethereal fluids' into the person being hypnotised, thereby changing your behaviour. A combination of phrenology and mesmerism meant that by touching a particular part of your head, say the 'tune organ' a mesmerist could supposedly improve a person's musical ability. Sounds uncomfortable and rather freaky.

Not surprisingly both phrenology and mesmerism were discredited by the mid 1800's. Not only because of the lack of scientific evidence but also perhaps because of the awful stereotyping and racism that was justified by some people, based on the size and shape of people's heads.

However, fast forward to the 20th century. Searching for a link between brain anatomy and psychiatric disorders is an area of intense research. Brain scans are seen as an important key to unlocking this puzzle. As people do different tasks, different areas

of the brain light up in the scanners' images. But brain scans are far more expensive than a pencil sketch of a Victorian head. So scientists are crowd sourcing their scans, sharing them across the world in the Enhancing Neuroimaging and Genetics through Meta-Analysis (ENIGMA) project, named after Alan Turing's code breaking WWII exploits. In this project the researchers aim to crack the genetic code, just as those at Bletchley Park used the first working computers to crack German military codes. The ENIGMA team collaboratively decide on the questions to be asked about psychiatric disorders and brain structures. Hundreds of scientists across the world are then sent an algorithm describing the steps to analyse the brain scans. They use the algorithms to analyse their scans and send back their results.

Ada Lovelace may have been worried that she might develop addictions, depression, or obsessive behaviors like her father, but her work in creating the first algorithm for the first computer has contributed to the technology now being used to study these disorders, and perhaps find a cure.

Billion of pounds are spent worldwide on alternative therapies such as reiki, and energy healing. Is this a modern day mesmerism or are we waiting for scientific evidence to prove they work? We may not have a hope organ but sometimes we are too hopeful about ideas with no evidence to back them!

Victorian volunteers needed

by Jane Waite, QMUL

**What was Ada Lovelace thinking about when she wrote:
“If amateurs of either sex would amuse their idle hours
with experiments on this subject, and would keep an
accurate journal of their daily observations, we should
have in a few years a mass of registered facts to compare
with the observation of the scientific”.**

Yes, crowdsourcing science experiments! Now we call it Citizen Science. She had just read a book by a Baron von Reichenbach on magnetism in which he had suggested a whole host of experiments, such as moving magnets up and down a person's body, showing people magnets in the dark, and holding heavy and light magnets and asking them if they felt any sensations. She could see that he had some great ideas, but she was not convinced by his examples alone.

Ada was not the only Victorian to ask the general public for help collecting data. Charles Darwin, the Origin of Species man, wrote to gardeners, diplomats, army officers and scientists across the world asking for information about the plants they grew and the animals (including people) they saw. This all helped him build up the concrete evidence that natural selection was the way evolution works. People even sent him gifts of live animals in the post. A Danish gentleman

sent him a parcel of live barnacles. When they did not arrive on time, Darwin, desperate to dissect the species, panicked and got ready to offer a reward in the *Times* newspaper. Luckily they arrived intact, fresh and not too smelly!

Today we might take part in the RSPB's Big Garden Bird Watch, contribute to a blog, 'favourite' a tweet, 'like' an Instagram post or vote for our favorite performer in a talent show. We participate, and 'amuse our idle hours' sometimes in the pursuit of science, sometimes not. Public research is a big new topic, with governments and companies looking to use people power. Innovations such as shared mapping systems ask users to upload details about a place, add photographs, rectify mistakes. Wikipedia is sourced by volunteers, with other volunteers checking accuracy. Galaxy Zoo volunteers even found a whole new planet that orbits four stars!

What would Ada be asking us to research? Test your own DNA and send in the results? Measure air quality and keep a record on a central database? Build your own 'find a barnacle' app? But rather than writing a journal or sending a parcel of barnacles, you would log it on line, click this link or design your own survey. Ada's computers are in on the act again.

Why not find a Citizen Science project on something you are interested in. Sometimes called public science or science outreach projects they might be run by local universities, museums, your council, charities or through crowdsourced internet projects such as www.zooniverse.org. Share what you do with others and spread Ada's word to be a modern day volunteer.

The Social Machine of Maths

by Ursula Martin, University of Oxford and Paul Curzon, QMUL



In school we learn about the maths that others have invented: results that great mathematicians like Euclid, Pythagoras, Newton or Leibniz worked out. We follow algorithms for getting results they devised. Ada Lovelace was actually taught by one of the great mathematicians, Augustus De Morgan, who invented important laws, ‘De Morgan’s laws’ that are a fundamental basis for the logical reasoning computer scientists now use. Real maths is about discovering new results of course not just using old ones, and the way that is done is changing.

We tend to think of maths as something done by individual geniuses: an isolated creative activity, to produce a proof that other mathematicians then check. Perhaps the greatest such feat of recent years was Andrew Wiles’ proof of Fermat’s Last Theorem. It was a proof that had evaded the best mathematicians for

hundreds of years. Wiles locked himself away for 7 years to finally come up with a proof. Mathematics is now at a remarkable turning point. Computer science is changing the way maths is done. New technology is radically extending the power and limits of individuals. “Crowdsourcing” pulls together diverse experts to solve problems; computers that manipulate symbols can tackle huge routine calculations; and computers, using programs designed to verify hardware, check proofs that are just too long and complicated for any human to understand. Yet these techniques are currently used in stand-alone fashion, lacking integration with each other or with human creativity or fallibility.

‘Social machines’ are a whole new paradigm for viewing a combination of people and computers as a single problem-solving entity. The idea was identified by

Tim Berners-Lee, inventor of the world-wide web. A new project led by Ursula Martin at the University of Oxford is working to make this a reality, creating a mathematics social machine – a combination of people, computers, and archives to create and apply mathematics. The idea is to change the way people do mathematics, so transforming the reach, pace, and impact of mathematics research. The first step involves social science rather than maths or computing though – studying what working mathematicians really do when working on new maths, and how they work together when doing crowdsourced maths. Once that is understood the project will be able to develop tools to help them work as part of such a social machine.

The world changing mathematics results of the future may be made by social machines rather than solo geniuses. Team work, with both humans and computers is the future.



Clever Genes

by Jane Waite, QMUL

Evolution was a hot topic in Victorian times. One person interested in whether parents passed on their traits and abilities to their children biologically was Francis Galton. He was particularly interested in how people became geniuses. There is no evidence to suggest that Francis Galton met Ada, but he did spend time with her husband after her death and included Ada and her father in his book 'Hereditary Genius'.

Galton, like his half-cousin Charles Darwin, author of the famous, ground-breaking book 'The Origin of Species', was interested in evolution and was trying to figure out whether there were connections across generations. But Galton's book did not connect flying squirrels and flying lemurs, or reptiles and birds, he was interested in eminent 'men of his time' and was looking for patterns in their families. He classified judges, statesmen, men of science, musicians, painters and poets scoring them according to how gifted members of their family were. Ada's father, Byron was classified as having 'hereditary ability' as Galton judged him to have eminently gifted relations, Ada!

The ideas driving Galton's attempts to draw conclusions on whether intelligence is inherited are still rumbling along, with researchers still using family studies, particularly with twins, to investigate it. Many conclude that intelligence is one of the most heritable traits, however there is still lots of controversy in this area of research. One of the problems is that with family often

comes not only genes but also wealth and opportunity, so the children of successful people have massive advantages in most societies irrespective of genetics.

New software, GCTA, Genome wide Complex Trait Analysis is providing a new way of looking at things, it estimates genetic influence in large samples of unrelated individuals. Complex algorithms churn through huge amounts of genetic data to find patterns related to traits such as height, genetic disorders and intelligence. The hunt is on for a clever gene!

Galton thought that Ada's mathematical genius was linked to her father's poetical genius. Perhaps her legacy will be to have contributed to the development of computer science and eventually the algorithms and programs that will reveal the sequence of DNA, that links their intelligence for real. Let's vote to name it the Ada gene. On the other hand perhaps the computers will show there is no such gene once and for all. Most likely we will find intelligence comes from a complex mix of genetics and upbringing.



One thing is clear, irrespective of genetics, anyone can be successful at anything if they are single-minded enough. The amount of effort you put into practicing skills matters far more than any innate advantages you may or may not have in your genes. So whether it is playing the violin, playing football or writing programs, by the time you have practiced for 10,000 hours with support from a good teacher, every one will be proclaiming that you are a born genius.

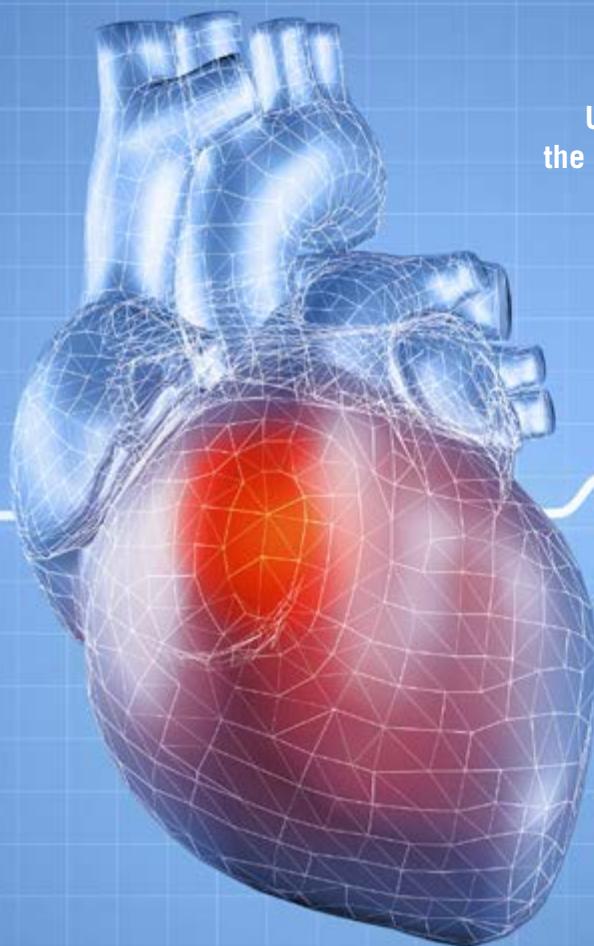
As a child Ada had a pet cat called Puff. In letters to her mother she referred to Puff as her mother's "granddaughter"!

"Your granddaughter has taken up all her kittens into a very nasty dirty hole in the roof of the house when nobody can get at them, she stays with them all day long and only comes down for her meals, I suppose their bed is made of cobwebs, and I think that Puff cannot have a very refined taste.

... Your very affectionate Carrier Pigeon A Ada Byron"

Understanding matters of the heart

by Paul Curzon, QMUL



Ada Lovelace mused that one day we might be able to create mathematical models of the human nervous system, essentially describing how electrical signals move around the body. The University of Oxford's Blanca Rodriguez is interested in matters of the heart. She's a bioengineer creating accurate computer models of human organs, so is working on a version of Ada's idea.



***The heart is
a working,
beating thing
not just a
sculpture***

How do you model a heart? Well you first have to create a 3D model of its structure. You start with MRI scans. They give you a series of pictures of slices through the heart. To turn that into a 3D model takes some serious computer science: image processing that works out, from the pictures, what is tissue and what isn't. Next you do something called mesh generation. That involves breaking up the model into smaller parts. What you get is more than just a picture of the surface of the organ but an accurate model of its internal structure.

So far so good, but it's still just the structure. The heart is a working, beating thing not just a sculpture. To understand it you need to see how it works over time. Blanca and her team are interested in simulating the electrical activity in the heart – how electrical pulses move through it. To do this they create models of the way individual cells propagate

an electrical system. Once you have this you can combine it with the model of the heart's structure to give one of how it works. You essentially have a lot of maths equations. Solving the equations gives a simulation of how electrical signals move from cell to cell.

The models Blanca's team have created are based on a healthy rabbit heart. Now they have it they can simulate it working and see if it corresponds to the results from lab experiments. If it does then that suggests their understanding of how cells work together is correct. When the results don't match, then that is still good as it gives new questions to research. It would mean something about their initial understanding was wrong, so would drive new work to fix the problem and so the models.

Once the models have been validated in this way – shown they are an accurate description

of the way a rabbit's heart works – they can use them to explore things you just can't do with experiments. They can explore what happens when changes are made to the structure of the virtual heart or how drugs change the way it works, for example. That can lead to new drugs.

They can also use it to explore how the human heart works. For example, early work has looked at the heart's response to an electric shock. Essentially the heart reboots! That's why when someone's heart stops in hospital, the emergency team give it a big electric shock to get it going again. The model predicts in detail what actually happens to the heart when that is done. One of the surprising things is it suggests that how well an electric shock works depends on the particular structure of the person's heart! That might mean treatment could be more effective if tailored for the person.

Balls, beams and quantum computers

by Jane Waite, QMUL

Have you played the seaside arcade game where shiny metal balls drop down to ping, ping off little metal pegs and settle in one of a series of channels? After you have fired lots of balls, did you notice a pattern as the silver spheres collect in the channels? A smooth glistening curve of tiny balls forming a dome, a bell curve forms. High scores are harder to get than lower ones. Francis Galton pops up again, but this time as a fellow Victorian trend setter for future computer design.



Francis Galton invented this special combination of row after row of offset pins and narrow receiving channels to demonstrate a statistical theory called normal distribution: the bell curve. Balls are more likely to bounce their way to the centre, distributing themselves in an elegant sweep down to the left and right edges of the board. But instead of ball bearings, Galton used beans, it was called the bean machine. The point here though is that the machine does a computation - it computes the bell curve.

Skip forward 100 years and 'Boson Samplers', based on Galton's bean machine, are being used to drive forward the next big thing in computer design, quantum computers.

Instead of beans or silver balls computer scientists fire photons, particles of light, through minuscule channels on optical chips. These tiny bundles of energy bounce and collide to create a unique pattern, a distribution though one that a normal digital computer would find hard to calculate. By setting it up in different ways, the patterns that result can correspond to different computations. It is computing answers to different calculations set for it.

Through developing these specialised quantum circuits scientists are bouncing beams of light forwards on the path that will hopefully lead to conventional digital technology being replaced with the next generation of supercomputers.

Magic: Hunting the high seas

by Jason Davison, School teacher
and professional magician

Ada Lovelace and Charles Babbage worked to try to automatically calculate nautical charts. In this pirate treasure themed magic effect you're able to mysteriously guide your spectator's freely chosen moves over a map to find the hidden booty. Go to the cs4fn website (www.cs4fn.org/ada/) to find out how the magic is done.

Cartoons, comics and computer games

by Jane Waite, QMUL

In 2009 for Ada Lovelace day, a comic strip about Ada and Babbage was created, not quite 100% historically accurate but certainly in the spirit of Lovelace's love of science and mathematics. Her thrilling adventures in Victorian London have now become a graphic novel.



In her own time, Ada was captured as a demure and beautiful young woman in portraits and sketches that were shared in books about her father. Ada would have sat for hours to have her portrait drawn, but she would have known about quick draw cartoons. Newspapers and magazines such as Punch contained satirical cartoons of the day. They were very influential in the 1840's. Faraday was drawn in Punch, but Babbage and Lovelace didn't make it then. But now they are crime busting mathematical superheros in their very own alternate history of computing comic book.

Books, films, even a musical have been created about Ada Lovelace, but as we write the circle has not quite been closed. There are no computer games about Ada. But maybe you could change that.

Autopilot Ada!

by Jane Waite, QMUL



Ada Lovelace dreamt of flying when she was a little girl, creating fantastical 'Flyology' sketches of steam powered flying machines. She would have been amused: Ada now flies real planes, hurtling through the air, miles above the ground, keeping million of people safe as they whizz around the globe. How? The autopilot systems of planes across the world are written in a programming language called Ada! Without Ada we would fall from the skies, splat!

Ada (the programming language) was created in the 1970's as the standard language to program all systems created for the United States Department of Defence (DoD). Before then the DoD used hundreds of programming languages, but they wanted one standard language that they could ask all their developers to use. When thinking up a name for the new language, they didn't pick an American programmer, or a man, no they picked the first programmer, our very own eighteenth century English Countess, Ada Lovelace.

The story of how the programming language was created would have appealed to Ada. She was from high society, liked nice expensive things, and dabbled with gambling. She bet on horses and liked a good old competition, and that is just what the DoD did. They ran a very big, very expensive competition. The DoD challenged the tech companies of the day to make the most secure, the most reliable, the least likely to ever fail computer programming language.

The race was on. Four teams took part: the Red, Blue, Green and Yellow teams (Ada liked to use colours too to make things easier to understand). Each created a new language for the US government. Then 400 volunteers tested the languages and picked the two best. The two winning colours, spent more time improving their design, and after another phase of testing by the volunteers a winner was selected. The Green team!

But even after they won the competition the testing wasn't over. Evaluation is everything if you are to get things right. Just as Ada (the person this time) had read over Babbage's work and corrected and improved it, so the army of volunteers, read over the language design. Over 500 reports were sent in from 15 different countries suggesting changes to the new language. Then Ada (the language this time) was 'born'. Ada was the most expensive programming language ever developed and it took over 5 years to create the first official version.

Ada (the language) was not just used by the DoD, it took the computing community by storm, not just for planes, but also for air traffic control, satellites, running banks, controlling subway trains and sending rockets to the Moon! Yes, Ada would have been amused.

"Today I have been flying particularly well and I think you will really say I have much improved in that exercise!"

– Letter from Ada to her mother

The Decline and Fall of Ada: Who's popular now?

by Jane Waite, QMUL

Ada (the language), is not the big player on the programming block these days. In 1997 the DoD cancelled their rule that you had to use Ada when working for them. Developers in commerce had always found Ada hard to work with and often preferred other languages. There are hundreds of other languages used in industry and by researchers. How many can you name?

Here are some fun clues about different languages. Can you work out their names? (Answers on the back page.)

1. A big snake that will squash you dead.
2. A famous Victorian woman who worked with Babbage.
3. A, B, __
4. A, B, __ (ouch)
5. A precious, but misspelled, thing inside a shell.
6. A tiny person chatting.
7. A beautiful Indonesian island.
8. A french mathematician and inventor famous for triangles.

Today, the most popular programming languages are, well we don't know, because it depends when you are reading this! Because what is fashionable, what is new is always changing. Plus it's hard to agree what 'the most popular' means for languages

(and pop stars!). Is it the most lines of code in use today? The favorite language of developers? The language everyone is learning? In July 2015 one particular website rated programming languages using features such as number of skilled software engineers who can use the language; number of courses to learn the language; search engine queries on the language and came up with the order. 1) Java 2) C 3) C++ 4) C# 5) Python. Where is Ada? 30th out of 100s! The same website had shown Ada (the language) as 3rd top in 1985! What a fall from grace.

But have no fear, Ada still survives and lives on in millions of lines of avionics, radar systems, space, shipboard, train, subway, nuclear reactors and DoD systems. Plus Ada is perhaps making a comeback. Ada 2012 is just being finalised, heralded by some as the next generation of engineering software with its emphasis on safety, security and reliability. So Ada meet Ada, it looks like you will be remembered and used for a long time still.

Github is a place where lots of programmers now develop and save their code. It encourages programmers to share their work. A kind of modern day, crowd sourced 'mass of shared facts' but coders would probably not say they did this just to 'amuse their idle hours'. Popular coding tools on this platform are JavaScript, Java, Python, CSS, PHP, Ruby, C++. Ada doesn't really feature, well not yet.

Looking to the Future

by Paul Curzon, QMUL

Ada Lovelace could see their potential long before a working computer was created.

- She realised that they could be used to manipulate more than just numbers and so their potential was more than as merely a calculator. They now manipulate numbers, words, pictures, music, films and more.
- She suggested they would one day be creative and be able to compose music. Programs that compose original music already exist.
- She thought that puzzles like peg solitaire might be represented mathematically in a way that computers might be able to solve them. Computers can solve all sorts of puzzles.
- She thought about how computers might be able to play games like Noughts and Crosses. It is one of the earliest games that artificial intelligence programs were successfully written to play. Now they can beat humans at virtually every game we ever invented.
- She suggested computers would one day be able to do algebra and weave "algebraical patterns just as the Jacquard-loom weaves flowers and leaves". We now call them theorem provers and they are used to help check complex software and hardware works.
- She thought computers might work on mathematical problems coming up with results that no human had discovered. Programs are now commonly used by mathematicians and have proven new results.



Back (page) in the air

by Peter McOwan, QMUL

Illness meant Ada spent much of her childhood confined to bed, but she put the time to productive use as she imagined ways to allow her to fly. She studied bird flight, chose materials to build her flying machine from, and even came up with an idea to use steam engine technology to power the flight. It's fitting therefore that many of the advances in flight today are powered not by steam, but by advances in computing that she helped establish.

No pilot needed

Computers now can fly planes on their own. It's perfectly possible for a jumbo jet to take off from London and land in New York safely without a pilot doing anything. The autopilot computer can do it all. Modern fighter jets are only kept in the air by the computer – a human couldn't react fast enough. To make them maneuverable the wings are smaller which means without the computer's constant adjustments they would just drop out of the sky.

Flight of fancy – Flies like a brick



cs4fn is edited by Paul Curzon, Jane Waite and Peter McOwan of Queen Mary University of London and Ursula Martin of University of Oxford. Autumn 2015. EPSRC supported this issue through research grant (EP/K040251/2). The issue was also supported by the Mayor of London and Department for Education. cs4fn is a partner on the BBC's Make It Digital Programme. cs4fn is generously supported by Google.

Answers: 1) Python 2) Ada 3) C 4) C# (C sharp) 5) Perl 6) Smalltalk 7) Java 8) Pascal

