Computer science

A conversation with Tristan Kirkpatrick

Tristan Kirkpatrick is the Director of Computer Science at Outwood Grange Academies Trust. He is also the Senior EdTech Project Lead for the trust and oversees the EdTech Demonstrator Programme and Outwood EdTech. Recently, Tristan gained 'Certified Innovator' status from Google, and has continued to contribute to the professional teaching community with EdTech solutions.

Website: www.tristank.co.uk

Twitter: @tristankp

In computer science, what does success look like by the end of Year 9 in terms of what students know, understand and can do?

In computer science and computing, you are aiming to teach the fundamentals really well: sequence, selection and iteration are the cornerstones of the subject; you would want all Year 9 students to be experts in sequencing, to understand and be able to apply selection in different contexts, and to understand and be able to use iteration, as that is quite a difficult concept to grasp. Beyond the cornerstones, you want to inspire a spark in young people, and computer science is a great subject for provoking sparks: it is so broad in its application, and at the same time, the concepts underpinning it are quite narrow. The

broad applications of those key concepts can be a real benefit when engaging students – you see the spark in students when they develop a program that moves something, or when they build a website that sells something. And that is the spark that you are looking for. If at the end of key stage 3 computer science students understand the cornerstones well, and have an enthusiasm for computer science, then that constitutes success.

You commonly teach programming, using the concepts of sequencing, selection, and iteration:

- Sequencing is putting things in order. In primary school, students will put instructions in order. They often use little robots and you press three arrows forward and it moves three paces forward and then it turns right and then goes forward again. And generally, you aim it at a chair leg at the other side of the classroom. So sequencing is putting steps in order. You can then apply that across the whole computer science curriculum, with varying levels of difficulty and complexity. When students come to write full programs, they are still using that skill of sequencing, of putting things in order.
- Selection is in terms of traffic lights: so if it is green, then the program can proceed, or if it is green then the robot can go; if it is not, then it cannot. It is that selection process that allows you to have different pathways through a program. Again, that is a concept that you teach all the way through the computer science curriculum, from making that robot move in primary school all the way through to writing huge programs at A level.
- Iteration is where you write one line of code and make it happen ten times. So you could instruct that robot go forward ten steps and it will go forward ten steps, rather than having to click the arrow ten times to move it forward.

Those are the key underpinning concepts of computer science. Alongside sequence, selection and iteration, there are other fundamentals that are important throughout the computer science curriculum:

- Abstraction: breaking down a problem to its simplest form;
- Decomposition: taking apart all the bits of a problem;
- Pattern recognition: recognising potential patterns and mathematical routes to do things;
- Writing algorithms.

Abstraction, decomposition, pattern recognition and writing algorithms are known as the four cornerstones of computer science. Most people want to know what app is being used, but that is irrelevant because you teach the concepts of sequence, selection and iteration; you do not teach how to write Python code or Javascript code. You teach the underlying fundamentals of those programming languages when you teach programming. Whilst you would write a series of instructions in the Python programming language that would move a robot across a room, you could also write a sequence of instructions that moves a robot across a room in Javascript. It is being able to transport the fundamental knowledge, understanding and skill of applying the concepts of sequence, selection and iteration between languages that makes the app, or whatever you are using, relevant. To begin with, it is all about those underpinning fundamentals.

Some of the best examples of lessons in programming are the ones using mini-whiteboards for checking and understanding. You can begin a programming lesson by asking students to write a sequence on how to make a cup of tea. Computer science is not just about programming, it is also about problem solving and being able to break a problem into a series of steps. That is why computer science is so important for students, because what they learn is so transferable to lots of other subjects, even down to being able to use abstraction and decomposition to prioritise the revision they need to do for their GCSEs. Essentially, computer science is about being able to think logically – computational thinking.

If computer science is taught thoroughly, it will prove difficult – especially programming and problem solving. You are always trying to build resilience in students and that takes time. At the end of Year 9, you want your computer studies students to be resilient learners because actually computer science is all about resilience and trying and trying and trying again. There are few subjects where 'failing' in an activity is a fundamental part of getting to the solution. There is a 'fail, fail, fail,

succeed' cycle that you need to build into students, especially at key stage 3.

So where do you begin in Year 7 if you are going to develop confident and competent computer science students by the end of Year 9?

You always, always begin with E-Safety, followed by a series of lessons on computational thinking. Part of the reason why students say this is hard is the new vocabulary: you have to start off with the fundamentals of pattern recognition, abstraction and algorithms, which is quite daunting for students. An effective pattern recognition lesson could look something like this: ask the students to draw on their whiteboards something that has a mouth, two ears, a nose and whiskers. And they draw something, and nearly every time they draw a cat. Actually what you emphasise is that they have not been told to draw a cat; all they have been asked to draw is a mouth, two ears, a nose and whiskers. You very occasionally get a student who draws some whiskers in one corner, a mouth in another etc., but most of the time you get a whiteboard full of cats. As humans, we recognise patterns and computers do not recognise patterns. So that is how you start with pattern recognition. Then you talk about abstraction. It helps to remove it from computers by talking about maps, for example. Give the students a satellite map of London and then give them a tube station map. You can then ask, 'Why don't we have all those trees on the London underground map?' It is because they are irrelevant to the problem we are trying to solve. So that is where you start, and students really enjoy it. They enjoy thinking differently. But for some students, it takes time. It is not natural for them to think like that, and you have to take small steps, especially in the beginning.

When it comes to programming, the first time they touch computers they use micro:bits, tiny little computer devices with just some LEDs on them and an LED screen. They are standalone devices. It is best to use Blockbuilder for programming, so rather than students having to worry about syntax errors and rogue commas and that sort of thing, students can drag and drop blocks onto the screen.

There is never enough time to cover everything in computing that you would like to cover. There is, within the computer science curriculum, a really hard theory element which is preparatory for later years. And quite

often you cover theory in key stage 3 and students are not absolutely clear they are learning computer science theory; as far as they are concerned, they are just solving problems.

Partnering with industry

Computer science is a brilliant subject for industry partnerships. There's a processor company called Arm Ltd: they make 'all' the mobile phone processors – way more than you would ever think – and we at Outwood Grange Academy Trust (OGAT) have linked with them. They come to school and do a full experience day, where students use the knowledge and understanding they have learnt from micro:bits in Year 7 and solve problems. The first time they worked with us was, quite honestly, the best day in my career because of the students' high levels of engagement. There is also a brilliant programme we are involved in called the First Tech Challenge, which is basically like *Robot Wars*, where students build their own robots, which is great fun. It fully supports every aspect of our curriculum. There is a serious message at the end of all of this: these are real careers. Students who enjoy this work can pursue a career in this field. There is quite a large push from CyberFirst, GCHQ and cyber security in recruiting students onto scholarships. That has been a real eye-opener for students when they realise that there are genuine opportunities at the end of studying computer science. We know just how important getting key stage 3 right is, to light that spark in students for computer science.

The computer science, IT and digital literacy curriculum

We begin text-based programming after February half term in Year 8. Building up to that point, there is an opportunity to do programming earlier on in a unit using Small Basic programming. In the same way that Python's a programming language, Small Basic is a programming language. And the reason we start with the Small Basic is that it fills in some gaps in coding for students as they type in their code. It is an assisted way of coding. We get students moving a turtle around on a screen, so that it draws shapes. The students might draw a square by going forward 100, 90 degrees right, forward 100, 90 degrees right, forward 100, 90 degrees right, forward 100. And then they would have drawn a square. They would screenshot that as their evidence. They are

working at the most basic level, but the same fundamental concepts – sequencing in this case – are used.

Programing begins with for real with Python. This is a programming language, which means that if we type in certain commands into the Python interface we can press play and run those commands and see what happens. The simplest command is, say, 'print Tristan Kirkpatrick' and it will print out your name. Programming is quite a heavy skill-based activity for which we provide a lot of scaffolding for students throughout the rest of the teaching. We scaffold each tiny step until students have the confidence to keep going further, gradually building upon their knowledge and understanding, progressing towards independence. For example, if the lesson is outputting data, which will be printing to a screen, we try to ensure that all students get to a point where they can output data. Obviously we may have some students who are quickly outputting data, so that they can progress and output data on two different lines. We might ask them to output first their name, then their age, then their date of birth, whatever it is, and we come back to that and discuss sequencing. So we have a few lessons of outputting data and students getting to grips with opening the Python interface, typing in the commands, understanding that if they get one letter wrong. In some instances it is teaching them how to use some of the functions of a keyboard at key stage 3. And then we move on to things like the concept of a variable which is tricky. They use the concept in mathematics and they would refer to it as X in mathematics, but actually in computer science, X is an unhelpful label for a variable because we do not know what it means. If X is supposed to stand for Age then in computer science we would just call it 'Age.' So then students might start developing simple programs where they type in their name, it prints their name out and then it stores their age, as the variable name, 'Age', and then it prints out 'Age'. So it is whatever you typed in as your Age. So we build and build, so at that point we are still sequencing.

Then we move into selection, which is where more exciting things start to happen with Selection, because you can then start asking a question: 'What is the capital of England?' And if they type in London with a perfect capital letter as a response, then it will say 'Great, well done, you typed in London, that is the correct capital'. And if they did not, it will

say, 'You got that question wrong'. So that is where we can start building that skill of Selection. Iteration is a challenge. Iteration, especially in Key Stage 3, is a really challenging concept to understand: we begin talking about looping through data and things happening more than once, just for one line of code. Quite often, we have to resort to physical models and perform the iteration on boards and draw looping arrows on bits of code. And that is really where we want to get to in Year 8.

In terms of programming in Year 9, we get to a point where students are given an algorithm problem – something like, 'A factory is making teddy bears; they make ten a day; you need to make a calculator where you can type in different scenarios: e.g. If five workers are working over this many days, how many teddies are going to be made?' And then they would go away, they would plan, they would create, they would test their solution. So we do that over a series of lessons: it is all scaffolded, and it simulates the kind of mini-project that a lot of the further learning beyond Year 9 requires. To be able to tackle that kind of task, students have to have resilience built into their ways of working.

The Design Technology process is a similar process: You get a brief; you decide how you are going to solve it; you solve it; then you evaluate it. It is exactly that kind of process we start applying to creating algorithms, so it does take a good while for us to develop students' ability, to a point where they can look at an algorithm and then undertake the full process from planning to evaluation.

Computer science is not all about programming; whilst programming is really important, at the same time the overarching computational thinking and understanding of how computers work are crucial to give the full picture of what students learn.

At OGAT, the majority of our curricula in our schools are a balance between ICT and computer science, but we have also begun embedding digital literacy. Now digital literacy you could call IT, but we differentiate computer studies, IT and digital literacy quite separately. In our core curriculum, I would say the makeup is about 50:50 IT/computer science. We still teach spreadsheets purposefully because we feel learners need to be prepared to be able to use spreadsheets, as there are so many jobs that require spreadsheet use. We also have a full theory unit, so we look at binary conversions and how a computer performs those conversions; hexadecimals; how computers display images. There is a strong theory

element, and we like that because it shows students that computer science is not just programming.

We recently had this problem: it had been identified that the life expectancy of the local population in the community of one of our academies was significantly below the national average. The school made what I feel is an incredible decision: to build a new curriculum based upon improving the life expectancy of its community. It is called the 'Curriculum for Life'. It aims to extend the life expectancy of the students in that school. And I always think of it as soon as we talk about curriculum; it is the first thing that springs to mind, if I am honest. So, we sat down, and we said, 'OK, within ICT and computer science, what do we want students to know and understand when they leave the school that they don't know and understand now?' It is a profound question. In response, we built a full course called digital literacy. It brings in history and financial planning, for example, but they are by-products of what we are teaching. Although it is not in the ICT curriculum (nor is it in the computer science curriculum, and it is certainly not on the examination specifications), it is equally important that we get that right for the people in that community. And what we have found is that the strength of some of that has begun to influence a number of schools' curricula. Colleagues have seen it and said, 'We don't necessarily have the life expectancy problem in our school, but that digital literacy unit – it really prepares students for life.' It has been a satisfying curriculum development, which you would not have expected in a world where P8 scores and examination outcomes are king.

Looking ahead, artificial intelligence is going to be huge; an understanding of artificial intelligence will begin creeping into the curriculum. The ethics of computing are going to be more important: self-driving cars, for instance, are an ethical minefield. Which way should you have a self-driving car go if it is about to hit four teenagers, and or a group of six grandparents? Which way should it go? Should it deviate off course? Should it not? Should it stick to its course?

There are clearly much better IT curriculum roots emerging, which is important, since it was devalued by the EBacc in 2010. That was something that was providing a genuine skill base to the sector. And ultimately, across the whole school, a bigger and better awareness of computer use safety will emerge.

Upskilling your SLT line manager

One of the big things SLT line managers of computer science need to know straight away is that computer science has been one of those subjects of real change and turbulence. We began with IT teachers, often originally business teachers who were trained – hopefully – to teach IT. And then all of a sudden, computer science came along and schools wanted to deliver computer science, and those former business teachers who had retrained to teach IT were now faced with the challenge of learning and then teaching computer science. It was something new, they did not know what it was, and it is a very different subject. The computer studies curriculum has built up over probably seven years; seven years of iterative improvement. Quite early on, it was clear that you needed to help support heads of subject to build the curriculum because actually the professional subject knowledge was not necessarily secure. So it needed to be a curriculum that you could train staff to use at the same time. Computer science teaching is now at the stage where it has gone through this iterative cycle: staff have upskilled nationally in computer science and they are now able to be significantly more creative in how the subject is delivered as the classroom practitioners have become more confident in their own knowledge, understanding and skills. It takes professional bravery to overcome significant self-doubt to be open to the professional learning involved in teaching computer science.

For an SLT line manager, it is a difficult subject. Often it is a head of faculty who might be in charge of that department, but even then, the head of faculty might not have the knowledge of the curriculum. So especially at key stage 3, make sure that teachers are able to articulate the intent of their curriculum: 'Why are you teaching micro:bits, for example, towards the end of Year 7? Why is the first time that they do text-based programming in Year 8?' More than anything, having the right questions from SLT line managers will provide the kind of space for thinking that departments need, to make sure that they have got it right.

In computer science, how does a department, even if it thinks its curriculum is challenging and cutting edge, know it has developed a brilliant curriculum? Where is their validation if they are the ones validating it? Departments need to reach out to the National Centre for Computer Science or the STEM Learning Centre at York and ask them to critique their computer science curricula. That critiquing process

is something that is crucial to helping departments to develop their computer science curricula.

First and foremost, line managers should read the national curriculum, just to understand what it is that we are expecting students to learn in those lessons. And the examination specifications are important. They provide a direction. The latest specification for computer sciences provides the vocabulary students need to know. That vocabulary does not change: you still talk about computational thinking, abstraction, decomposition etc. from Year 7 all the way through to Year 11, and on to Year 13. If a line manager is overseeing a department and trying to challenge that department, making sure the right questions are asked when looking at a department's course plans is crucial. The SLT line manager will have to work hard to understand the fundamentals of computer science; it is a demanding and complex subject to challenge.

Computing: backgroud

It is worth quoting the purpose of computing in the national curriculum programme of study:

> A high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world. Computing has deep links with mathematics, science and design and technology, and provides insights into both natural and artificial systems. The core of computing is computer science, in which pupils are taught the principles of information and computation, how digital systems work and how to put this knowledge to use through programming. Building on this knowledge and understanding, pupils are equipped to use information technology to create programs, systems and a range of content. Computing also ensures that pupils become digitally literate – able to use, and express themselves and develop their ideas through, information and communication technology – at a level suitable for the future workplace and as active participants in a digital world.[1]

1 www.bit.ly/3gcVb4n

The national curriculum for computing aims to ensure that all pupils

> can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation; can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems; can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems; are responsible, competent, confident and creative users of information and communication technology.

Once the importance statements have been revisited, it is helpful for subject leaders and coordinators to discuss and agree with colleagues the reason why their subject, in this case computing, is important for the pupils in their school. One way of doing this is to draw on a quote, in this case from George Dyson: 'Alan Turing gave us a mathematical model of digital computing that has completely withstood the test of time. He gave us a very, very clear description that was truly prophetic.' This kind of prompt allows us to formulate our way of stating the importance of the subject. We might agree or disagree with such a statement and in doing so come to a form of words which expresses our view of the importance of this subject, in this school. This moves us away from the territory of 'We teach this subject because of the SATs or GCSEs.' While the external tests and exams are important, they are not the totality of the subject.

Professional communities

Subject associations are important because at the heart of their work is curriculum thinking, development and resources. The subject association for computing is Computing at School[2] and it should be the case that any member of staff with responsibility for a subject should be a member of the relevant subject association, and this should be paid for by the school.

2 www.computingatschool.org.uk

Twitter subject communities are important for the development of subject knowledge, because it is here that there are lively debates about what to teach, how to teach and the kinds of resources that are helpful. For computing it is worth following Computing at School[3] and the hashtag is #CASChat.

LINKS

Ten tips for teaching programming – www.bit.ly/3k5OIsZ
Computing at School – www.computingatschool.org.uk
Network of Excellence – www.computingatschool.org.uk/noe
Barefoot Computing – www.barefootcas.org.uk
Code Club – www.codeclub.org.uk
CoderDojo – www.coderdojo.com
RaspberryPi – www.raspberrypi.org/education
Scratch Team – www.scratch.mit.edu
Hello World: Magazine for Computing and Digital Making Educators – www.bit.ly/3k3oV4X
Computer Science Teachers Association – www.csteachers.org
Digital Schoolhouse – www.digitalschoolhouse.org.uk/resources

3 www.twitter.com/CompAtSch

An skeleton overview of the Outwood Grange Key Stage 3 computer science curriculum

	Year 7	Year 8	Year 9
Autumn	• Internet safety • ICT: Spreadsheets • ICT: Word Processing • ICT: Presentations • CS: Computational Thinking	• Digital Literacy introduction • CS: Small basic • DL: Creative task/ assessment	• CS: HTML web design • CS: Theory
Spring	• CS: Computational Thinking • CS: Using Micro:bit	• CS: Basic theory • CS: Programming	• CS: Programming • ICT: Photoshop skills
Summer	• CS: Using Micro:bit • ICT: Creative tasks	• CS: Programming • DL: Creative task/ assessment	• CS: Programming • CS: Digital literacy

Three documents for your senior leader line manager to read about computer science

1. The national curriculum
2. 'What is computational thinking?' – www.bit.ly/3k3DyFf
3. 'Policy briefing on teachers of computing' – www.bit.ly/3xXU4LK

Five questions for your senior leader line manager to ask you about computer science

1. How do you plan to build on the four cornerstones of computer science throughout your curriculum?
2. What is the balance between computer science, digital literacy and ICT within your curriculum?
3. What would success look like for your learners at the end of their school-life studying computer science?
4. Throughout a student's learning, what programming languages will they be exposed to and why have you chosen these?
5. What do you plan to do to increase the number of girls taking up computer science at GCSE?