MILES BERRY PRINCIPAL LECTURER IN COMPUTER EDUCATION

WHAT I'M THINKING ABOUT COMPUTATIONAL THINKING

More automated solutions, fewer jam sandwiches

Computational thinking is the golden thread running through England's computing curriculum, and has been a profoundly influential concept in computing curriculum design across the globe. There are, though, two quite different ways of thinking about computational thinking. The position you take on this isn't merely a philosophical one: it seems to impact directly on the 'what' and 'how' of computing education.

Firstly, there's the idea of taking the principles of computer science and applying them to other contexts. Jeannette Wing is credited with coining the term 'computational thinking'. Her original position, back in 2006 was:

"Computational thinking involves solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science."

In this view of computational thinking, we might identify a number of the concepts that computer scientists and software engineers draw on in their work, and then teach pupils about these concepts: thus pupils might learn about algorithms through writing recipes for jam sandwiches, they might learn about decomposition by labelling the parts of a plant, they might learn abstraction through planning journeys on the underground, and they might learn to recognise patterns through spelling rules. With this interpretation of computational thinking, unplugged approaches seem to work well.

For me, though, this definition of computational thinking seems too vague to be helpful. There seems little here to set computational thinking approaches apart from those which we might label as mathematical reasoning, or engineering thinking. Without computation, this sort of computational thinking is really just thinking.

I think there's a better way. Recently, Jeannette Wing has given a more helpful definition of computational thinking, which sites it within the broader domain of problem-solving but, crucially, links this explicitly to computation:

"Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer – human or machine – can effectively carry out."

I think the key distinction between computational thinking and other problem-solving approaches is that, with computational thinking, we should be searching for a solution that can be automated. Typically, this means that the solution should be one that can be implemented as code. By locating computational thinking as part of the programming process, it becomes a teachable, and assessable, thing. Computational thinking then becomes useful in solving the problems, or understanding the systems, that come up in the other subjects that students study. It is less about finding examples elsewhere in the curriculum to illustrate CS ideas, than about applying these ideas to write programs that solve problems from elsewhere in the curriculum.

While making jam sandwich recipes, labelling plant diagrams, planning tube journeys, and learning spelling rules are all well and good, they don't sit well with this view of computational thinking – in none of these cases do pupils take what they've done and implement it as code. Pupils are far more likely to learn computational thinking if they can use it to write programs to solve problems: you can't, for example, create Instagram-style filters, model the spread of an epidemic, or suggest a spelling correction without using computational thinking, and it's in looking for an automated solution that can be applied to any problem in the class that computational thinking becomes distinctive and useful. (HW)

MILES BERRY

Miles is principal lecturer in Computing Education at the University of Roehampton. He's a member of the Raspberry Pi Foundation, and serves on the boards of CAS and CSTA.