Mark Gadsby, 16th September 2019

How can the main aims of the secondary curriculum be met through the teaching and learning in your subject?

Computational thinking is often cited by computing educators as the gift computing offers to the general secondary curriculum. It is argued that it provides a framework of thinking that can be applied productively across subject boundaries. I am asserting that without computer programming underpinning computational thinking, what remains is an abstract framework for problem solving which looks suspiciously like just plain thinking. Notwithstanding that such a framework for problem solving has merit, compared to established evidence-based scientific methods of problem solving, its necessity appears questionable. I will argue that the main aims of the secondary curriculum may be met through computational thinking, but that it is necessary to acquire a solid understanding of computer programming before it can be effectively applied.

One of the key aims of the national curriculum is the acquisition of essential knowledge to unlock the opportunities of later life (Department for Education, 2014). Michael Young (2007) names new ways of thinking about the world as ‘powerful knowledge’. This concept was as important to the Chartists in 19th century as it is to parents today, especially those in developing countries who make sacrifices so their children can attend school. The modern world is increasingly a digital landscape with jobs unimaginable a few years ago now common place, and equipping students with the wherewithal to navigate and creatively contribute to that landscape is vitally important. Against this backdrop powerful knowledge is increasingly specialist knowledge; computer programming is an excellent example of such specialist knowledge.

For Tedre and Denning (2016), the assertion that computing equals programming is old fashioned, debunked, and a trap that will lead those caught to an overly narrow view of computing as a subject. For them a programming focus misses the riches of computer architecture, networks, design, and computational science. Furthermore, it will offer a boring and uninviting experience that is too analytical, abstract and far removed from the real world. However, in my opinion, an understanding of how computer programs are structured, along with their capabilities and limitations, is invaluable for understanding the digital representation of the real world. I have asked students what they find most appealing about the current computer science syllabus and the answer is always that the programming is most interesting. What the computer programmer does is develop abstractions of the machines capabilities to model and improve the real world in a practical and applied manner, an activity many find compelling and rewarding.

Computational thinking was first coined as a phase by Seymour Papert in his 1980 book Mindstorms. He took a very practical approach and championed leaning by doing, and that doing had programming at its heart. The term was subsequently embellished by Jennette Wing (2006) for whom abstraction was as important as programming. For her the key skill was being able to work at multiple levels of abstraction. Today the Computing at School framework for computational thinking (Kemp, 2014) certainly has abstraction at the forefront along with decomposition and algorithmic thinking; being able to recognise and generalise patterns is also included. My pivotal argument is that you can't teach abstraction without the reference point provided by the programming language. If students were to be taught a procedural programming language and then, ideally, a second language preferably from a different paradigm, they will be able to see and understand the relevant abstractions for themselves.

Let us apply my argument to decomposition. Take, for example, a real world problem that does not have an obvious computational solution, that of recognising an individual in a crowd. This is something humans (and other animals) can do seemingly instinctively, we have evolved a technique using a combination of our senses that is extremely effective. Decomposing this process is not helpful when thinking of designing a computational solution to this problem; computers don't have senses. Commonly, computer solutions work by comparing facial features with those in a database and this is not commensurate with the human process. Decomposing a problem is not always useful in itself; the problem must be decomposed down to steps that can be achieved by a computer. The capabilities and limitations of the computer are most thoroughly understood by learning computer programming.

Computer programming is the art of the possible but we do not lose the ability to think abstractly by focusing on it. Programming languages themselves are abstractions of the computers' architecture. Limitless further abstractions can be achieved by designing well structured and modular functionality, built into libraries and application programming interfaces. The important feature of this approach is that everything is underpinned by the code which remains the golden thread.

I agree that computing amounts to the fourth 'r', as imperative to education as English and maths (Wing, 2008) and I appreciate that computational thinking is not often presented without computer programming. The issue is that programming is often denigrated to a tool to be used for other ends and this misses a vital point; that it is a vocabulary we can use to express real world problems in terms that optimise computers' potential.

The elements of computational thinking have been practised by people for millennia, what studying computer science adds is a mechanism by which ideas can be applied within the modern digital landscape. Specifically, computer programming is the means by which these ideas can be applied and as such is an extremely powerful knowledge to learn, with numerous application across the curriculum.

References:

Department for Education, 2014. The national curriculum in England, Key stages 3 and 4 framework document. 2.1 and 3.1.

Kemp,P. 2014. Computing in the national curriculum, A guide for secondary teachers. Computing At School, page 5.

Papert,S., 1980. Mindstorms: Children, Computers, and Powerful Ideas. Basic Books.

Tedre, M. and Denning, P.J., 2016. The long quest for computational thinking. Proceedings of the 16th Koli Calling Conference on Computing Education Research (pp. 24-27).

Wing, J.M., 2006. Computational thinking. Communications of the ACM, 49(3), pp.33-35.

Wing, J.M., 2008. Computational thinking and thinking about computing. Philosophical transactions of the Royal Society of London A: mathematical, physical and engineering sciences, 366(1881), pp.3717-3725

Young, Michael. 2007. What are schools for? Educação & Sociedade. 28. 1287-1302.