Ofsted
raising standards
improving lives

Research and analysis

Research review series: computing

Published 16 May 2022

Applies to England

Contents

Print this page

Introduction

Digital technology is driving extraordinary global changes that some are calling the Fourth Industrial Revolution.[footnote 1] Navigating these changes effectively and safely requires a significant understanding of digital literacy, information technology and computer science. This knowledge is also crucial if business, industry and individuals are to exploit the opportunities offered by this revolution. The national curriculum makes it clear that computing is mandatory at key stages 1 to 4 and that 'a high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world'.[footnote 2]

This review explores the literature relating to the field of computing education. Its purpose is to identify factors that can contribute to high-quality school computing curriculums, assessment, pedagogy and systems. We will use this understanding of subject quality to examine how computing is taught in England's schools. We will then publish a subject report to share what we have learned.

The purpose of this research review is set out more fully in the 'Principles behind

Ofsted's research reviews and subject reports'.[footnote 3]

Since there are a variety of ways that schools can construct and teach a high-quality computing curriculum, it is important to recognise that there is no single way of achieving high-quality computing education.

In this review, we have:

- outlined the national context in relation to computing
- summarised our review of research into factors that can affect the quality of education in computing
- considered curriculum progression in computing, pedagogy and assessment, and the impact of school leaders' decisions on provision

The review draws on a range of sources, including our 'Education inspection framework: overview of research', which sets out the 3 phases of our curriculum research.[footnote 4]

We hope that, through this work, we will contribute to raising the quality of computing education for all young people.

Terminology

Many different terms are used in England, in the rest of the UK and internationally to describe computing education, such as computing, computer science and informatics. The research we have considered here uses all of these terms; however, for the purpose of this review, we will use the standardised term 'computing', as this is the name given to the subject in the national curriculum.

National context

Computing in the national curriculum replaced information and communication technology (ICT) in 2014. This followed successful representations by those in industry, academia and subject groups. Some felt that the ICT curriculum focused too heavily on office skills and did not allow pupils to develop knowledge that enables technical innovation.[footnote 5] The government's own consultation on the change highlighted negative views of curriculum content.[footnote 6] Larke observed that 'overall, a narrative of ICT as academically weak and vocationally useless prevailed'.[footnote 7] In contrast, computing education is considered to be important, because it has social, cultural and economic benefits. Through computing education, pupils can learn 'powerful knowledge', enabling them to become informed and active participants in our increasingly digital society.[footnote 8] Despite its importance, in 2017 the Royal Society suggested that 'computing education across the UK is patchy and fragile'.[footnote 9]

The national curriculum for computing established computing as a foundational discipline that every child studies. In England, the national curriculum states that pupils should study computing until the end of key stage 4.[footnote 10] All pupils in maintained schools are expected to study the national curriculum. Our handbook states that 'academies are expected to offer all pupils a broad curriculum that should be similar in breadth and ambition'.[footnote 11]

Reception and primary

Pupils' development of early computing knowledge is important. Grover, Pea and Cooper have suggested that:[footnote 12]

> " Learners' success in future engagement with computing will depend on how well introductory curricula prepare them in both the cognitive and affective dimensions of computational learning."

Computing is not part of the latest statutory framework for the early years foundation stage,[footnote 13] but is part of the national curriculum from Year 1. Recently, there has been a debate on learning computing in the first years of schooling and the importance of getting it right.[footnote 14] Several studies have demonstrated that young pupils are able to wrestle successfully with the core concepts of computing, including more technical subject content such as programming and robotics.[footnote 15] That said, it is important that children experience teaching informed by expertise.[footnote 16]

The national curriculum sets out the content that primary school pupils should learn in computing.[footnote 17] A 2017 report by the Royal Society identified that primary-age pupils typically have 1 hour a week of computing education; however, the research informing the report noted that this varies and there are a small number of primary schools where pupils receive no computing education at all.[footnote 18] Three years after the new programmes of study for computing were introduced, the same research found that teachers saw computing as the 'future' and felt that there was a clear rationale for teaching the knowledge and skills required. Alongside these positive aspects it also found that:[footnote 19]

> " Primary school teachers unfavourable to the new curriculum described the requirements as being too advanced for the available physical resources and budget, that staff lack the required skill-set and knowledge to teach the subject, and that the language used in the curriculum is overly-technical."

The research also highlighted that the main obstacle to teaching computing faced by teachers was a lack of technical subject knowledge. An international study from 2019 found that many primary school teachers were concerned about their own personal subject knowledge and the resources available to teach the intended curriculum.[footnote 20]

Secondary

As in primary schools, the national curriculum sets out the subject content all pupils should learn in secondary school.[footnote 21] This includes provision for all pupils to study computing in key stage 4.

The 2017 report by the Royal Society highlighted that 1 hour a week of computing teaching was not adequate to teach the key stage 3 curriculum.[footnote 22] A recent report using government census data showed that the amount of curriculum time allocated to computing in key stage 3 fell from an hour to just over 45 minutes between 2012 and 2017.[footnote 23] The same report suggested that pupils in key stage 4 who are not studying a computing qualification receive little timetabled computing education. These findings suggest that not all pupils are receiving sufficient curriculum time to learn the computing subject content set out in the national curriculum.

There are a number of computing qualifications that pupils can complete at key stage 4. These include the computer science GCSE and a range of non-GCSE qualifications; however, the Royal Society highlighted concerns from teachers about the credibility of such non-GCSE qualifications and how well they cover the scope of subject content.[footnote 24]

An analysis of government statistics of examination data to understand pupils'

access to computer science qualifications in schools in 2018 showed that:[footnote 25]

- the number of schools offering GCSE computer science had increased, with nearly 80% of Year 11 pupils in schools that offered a GCSE in computer science
- state schools with higher proportions of disadvantaged pupils were less likely to offer a GCSE in computer science
- 7.6% of pupils were in a school that did not offer key stage 4 computing qualifications

This data highlights that, although access to computing education is improving for most pupils, there are still inequities in provision.

After computing was introduced into the national curriculum, there was a rapid increase in the number of pupils studying computing GCSE qualifications. A-level computing has also seen a sharp increase in the number of entries, which have more than doubled since 2016.[footnote 26] However, the number of pupils entering A-level computing is still much lower than the number entering subjects like biology, chemistry and physics. Despite the increase in the number of pupils studying computer science, the total number studying computing qualifications at key stage 4 fell substantially. This is in large part due to the withdrawal of ICT GCSE in 2019. [footnote 27]

Gender

Ten years ago, we reported on the gender imbalance in ICT, noting that the 'number of girls entered [for GCSE ICT] continued to lag behind boys and the percentage of girls entering AS and A level has remained static at around 35% of the cohort'.[footnote 28] This gender imbalance is significantly higher in the new subject of computing. Analysis of examination data from 2021 shows that there are disproportionately few girls in computer science: they make up only 21% of entries at GCSE and only 15% of entries at A level.[footnote 29] This data adds to a continuing trend of low entries for girls.

In 2017, the Royal Society reported that gender balance was 'the most significant diversity issue' in the subject.[footnote 30] A recent snapshot survey of 350 girls aged 14 to 18 provides some supporting evidence to this. Over a quarter of respondents said that the subject is boring and nearly a fifth said that they lack interest.[footnote 31] Some respondents also inferred confidence in their ability as a reason for why they did not choose computing for further study.

Other organisations have noted the gender imbalance in computing. For example, The Wellcome Trust found that, by Year 9, there was a large gender divide in the subject, with less than half the number of girls finding the subject interesting compared with boys. The same report noted that, from Year 7 to Year 9, computing was the least enjoyed subject for girls.[footnote 32] Other research has indicated that girls tend to underestimate their performance in programming compared with boys and are less confident in their capacity to succeed in computing.[footnote 33] Research has shown that, although girls outperform boys in computing, both girls and boys underperform in the subject compared with their achievement in other subjects. [footnote 34]

Recruitment/workforce

Teachers' content knowledge and pedagogical content knowledge are important factors in high-quality computing education. This is because this knowledge helps teachers decide, for example, what to teach, how to question students about it and

how to deal with problems of misunderstanding.[footnote 35] However, research consistently identifies that there is a lack of suitably qualified computing teachers to teach the subject.[footnote 36]

A 2017 UK-wide survey of teachers with responsibility for computing education found that only a small percentage of primary school teachers held a computer science qualification as their highest qualification.[footnote 37] While primary school teachers cannot be expected to hold specialist qualifications in all the subjects that they teach, this research highlights the importance of subject-specific continuing professional development (CPD) in primary schools.

In secondary schools, 46% of computing teachers held a computing qualification (36% computer science and 10% ICT or business with ICT).[footnote 38] In 2018 and 2019, just under half of the hours taught in computing in secondary schools were taught by a teacher with a relevant post-A-level qualification. This contrasts with other EBacc subjects, where most hours were taught by subject specialists.[footnote 39]

Recruitment of computing teachers had been consistently below target since 2014.[footnote 40] However, recent initial teacher training (ITT) census data shows that in 2020/21 recruitment targets for computing were met.[footnote 41] This represented a 30% increase over the previous year.[footnote 42] However, this sits alongside a 23% increase in new entrants to ITT for all subjects, so may not be significant in the long term.[footnote 43] A 2019 policy briefing drawing on government data highlighted that the drop-out rate for teachers in ITT is higher in computing than in most other subjects.[footnote 44] It also draws attention to research that identifies the salary differentials between teaching and other areas of employment in computing.

Curriculum

This review explores the forms of knowledge that pupils need to learn in order to make progress in computing. Ofsted's education inspection framework (EIF) and the research underpinning it are the lenses through which we have selected, considered, framed, presented and connected relevant research and critical commentary. We have also used findings from our overview of research and applied these critically to arrive at a conception of a high-quality computing education.[footnote 45] This is particularly the case for areas where there is limited research on the nature of a high-quality computing education.

Pillars of progression

A useful way of thinking about progression in computing is to consider the 3 main content areas that pupils develop knowledge of:

- computer science
- information technology
- digital literacy

These 'pillars' of progression are recognised as areas of the curriculum by the Royal Society and are visible in the aims of the national curriculum for computing.[footnote 46] Pupils make progress in computing by knowing and remembering more about and, importantly, across each of these categories,[footnote 47] and being able to apply this knowledge. However, these pillars do not sit separately from each other. Knowledge from each pillar complements the others and some subject content only exists at the interplay between these 3 pillars.[footnote 48]

Declarative and procedural knowledge

This review draws a distinction between declarative and procedural knowledge in computing.[footnote 49] Declarative knowledge, often referred to as conceptual knowledge in the literature, consists of facts, rules and principles and the relationships between them.[footnote 50] It can be described as ‘knowing that’. In contrast, procedural knowledge is knowledge of methods or processes that can be performed. It can be described as ‘knowing how’. Examples of declarative and procedural knowledge across the 3 pillars can be seen in Table 1.

Table 1: Examples of declarative and procedural knowledge in computing

Form of knowledge	Computer science	Information technology	Digital literacy
Declarative	Programming syntax The purpose and function of different logic gates	Principles of effective multimedia design Spreadsheet formulae	Features of unreliable content
Procedural	Performing binary addition Implementing a repeat in a programming language	Setting up a slide master Applying conditional formatting	How to perform an advanced web search

This distinction is helpful when considering knowledge within the subject. Many aspects of computing use skills such as programming, creating digital artefacts and being able to use a search engine. It is helpful to consider these skills in terms of procedural knowledge, as they are methods and processes that can be performed. This makes identifying the knowledge required to perform these processes skilfully much easier. They are enabled by declarative knowledge such as knowledge of suitable data types and structures, knowledge of appropriate font sizes and styles and knowledge of suitable key words to use when performing searches.

Based on the above, high-quality computing education may have the following features

- The planned curriculum includes a breadth of knowledge relating to computer science, information technology and digital literacy.
- Declarative knowledge (‘knowing that’) and procedural knowledge (‘knowing how’) are identified, sequenced and connected in the curriculum.
- Skilful use of technology is underpinned by procedural and declarative knowledge.

Computer science

Computer science covers knowledge of computers and computation, including concepts such as data, system architecture, algorithms and programming. Computer science is seen as the core of computing and underpins the whole of the subject.

[footnote 51] Because of this, it is fair to say that computer science provides the foundational knowledge required to understand and interpret the other areas of the computing curriculum. Therefore, it is important that any computing curriculum is rich in computer science knowledge.

Programming

Programming is an important part of the computing curriculum. In the national curriculum, it appears throughout the programmes of study for computing. It allows pupils to apply their knowledge of computer science through writing code to solve problems. Peyton Jones describes programming as:[footnote 52]

> “ …the lab work of computer science: it motivates, illuminates, and brings to life the dry bones of theory. Without programming, computer science would be a dry, theoretical husk of a subject. Imagine a music lesson where the students only studied the rules of counterpoint or the structure of sonata form, but never brought them to life by performing or composing such music!”

Building programming knowledge

Learning to program is considered to be difficult.[footnote 53] When pupils are learning to program, they are often expected to learn different things at the same time, for example:

- programming constructs such as sequence, selection and repetition
- programming language syntax and semantics
- how to solve problems using programming

Programming is seen primarily as a skill that pupils develop. However, Schulte notes that learning to program successfully involves learning a body of knowledge,[footnote 54] including knowledge of:

- programming languages
- tools like compilers and development environments
- programming styles
- standardised solutions to programming problems

A literature review from 2010 to analyse programme comprehension models concluded that the ‘role of domain knowledge for comprehending programs seems to be underestimated’ and that ‘novice programmers’ early comprehension models can be characterized by a pattern of “holey knowledge” like an incomplete patchwork quilt’.[footnote 55] That is to say that novice programmers have a fragile and incomplete mental model for comprehending programs. This analogy of how pupils develop programming knowledge puts into context a generic finding that was foundational to the EIF, namely the role of schema, or structures of memory that build and link knowledge over time. Experts have more detailed schema than novices.[footnote 56] The problems that novices face in programming arise in part from a lack of organised knowledge.[footnote 57]

Programming misconceptions and the notional machine

Teachers might be tempted to expect pupils to write code at the very early stages of their programming education, before they know what that code will do. Du Boulay describes this as:[footnote 58]

> “ Learning to program is like learning to use a toy construction set, such as Meccano, to build a mechanism, but as if inside a darkened room with only very limited ways of seeing the innards of one’s creation working.”

Through this limited view it is easy to see how pupils might develop misconceptions. Sorva notes that pupils’ programming misconceptions have some features in common. For example, pupils ‘commonly lack a viable model of program execution.

In other words they fail to understand the notional machine they are learning to control'.[footnote 59]

A notional machine is an abstracted mental model of how a program will be executed within a programming language. Put simply, it is the knowledge of what a program will do when it is run. The national curriculum expects pupils to 'use logical reasoning to predict the behaviour of simple programs'.[footnote 60] Building this mental model or notional machine can be made possible through pupils developing knowledge of what the code they write will do. If pupils do not develop a secure mental model of how their programs will execute, they are more likely to make incorrect inferences and may develop misconceptions.[footnote 61]

Programming language choice

A core element in the debate about programming in the curriculum and how to teach it is the choice of programming languages. The national curriculum does not prescribe programming languages beyond the need for 2 languages to be taught in key stage 3, at least one of which should be textual.[footnote 62]

It is common for block-based languages such as Scratch to be used in primary schools and to a lesser extent in secondary schools.[footnote 63] Block-based programming languages can be useful in teaching programming, as they reduce the need to memorise syntax and are easier to use. However, these languages can encourage pupils to develop certain programming habits that are not always helpful. For example, small-scale research from 2011 highlighted 2 habits that 'are at odds with the accepted practice of computer science'.[footnote 64] The first is that these languages encourage a bottom-up approach to programming, which focuses on the blocks of the language and not wider algorithm design. The second is that they may lead to a fine-grained approach to programming that does not use accepted programming constructs; for example, pupils avoiding 'the use of the most important structures: conditional execution and bounded loops'. This is problematic for pupils in the early stages of learning to program, as they may carry these habits across to other programming languages.

Further research has highlighted that, although block-based languages may help novices to overcome the difficulties with syntax that they can face when learning to program, they do not necessarily help pupils with the semantic and conceptual difficulties.[footnote 65] It is therefore important that, if schools use block-based languages, they consider how to design the curriculum to mitigate these potential pitfalls.

It would seem logical that the knowledge pupils need in order to make progress should dictate the programming languages that schools choose for them to learn. However, this is not always true. A 2019 pilot study of curriculum implementation across 7 countries found the opposite: 'Interestingly we expected that curriculum would drive teachers' motivations for selecting programming languages, however, our results discovered this isn't the case and that student-driven factors motivate selection'.[footnote 66] These factors include teachers' perception of how appropriate the programming language is for the pupils' age. Over half of the respondents in the 2019 study selected visual programming languages for pupils in all secondary year groups. Cost and availability were other reasons for choosing programming languages. Preferences of leaders and the demands of the curriculum appeared to be less influential. Research indicates that the choice of programming language is important and should be considered carefully.[footnote 67] The EIF highlights the importance of selecting resources to match the ambitions of the curriculum.[footnote 68]

Computational thinking and problem-solving

When pupils solve problems in computing, this is often described as computational thinking (CT). CT has gained great prominence in computing education. Reference to CT can be found in the national curriculum and in GCSE exam specifications.[footnote 69] CT is a term stemming from Papert's work in the 1980s;[footnote 70] more recently, it

has been popularised as a ‘fundamental skill for everyone’.[footnote 71] This is often used to justify including computing in the curriculum. However, in their discussion of the role of CT, Tedre and Denning point out that we should be cautious about the scope of CT and avoid exaggerated claims that it develops problem-solving skills that are transferrable to other domains.[footnote 72] Learning computing as a subject and the knowledge related to CT is a worthwhile pursuit in itself, and does not need to be justified with tenuous claims about broader benefits.

CT is uniquely conceived, and the literature on this subject uses a range of different definitions. It has been claimed that it is important to have a clear definition of CT and that teachers may ‘struggle with the various and conflicting interpretations of its nature’.[footnote 73] As part of a project working with experts to define CT, Barr and Stephenson state that CT is, in part, an ‘approach to solving problems in a way that can be implemented with a computer’.[footnote 74] This definition makes designing the curriculum to develop CT more straightforward, as the required declarative and procedural knowledge can be identified and sequenced more easily. There are widely accepted core elements of CT that can be used to form a curriculum.[footnote 75] Grover and Pea have produced a list of concepts and practices related to CT:[footnote 76]

- logic and logical thinking
- algorithms and algorithmic thinking
- patterns and pattern recognition
- abstraction and generalisation
- evaluation
- automation

The above list sets out areas of problem-solving in computing; however, it does not describe how pupils become better at problem-solving in these areas. These areas do not develop through a ‘learned skill’; Tricot and Sweller argue that teaching generic skills does not work and ‘learned skill, especially problem-solving skill, derives primarily from the accumulation of a large store of domain-specific knowledge stored in long-term memory’.[footnote 77]

A literature review carried out for the 2017 Royal Society report highlights this work on the role of domain-specific knowledge in problem-solving and links this to other areas explored later in this review, such as the use of subgoals.[footnote 78] CT can be seen as difficult to teach,[footnote 79] which may be because there is no clear definition of it, or a belief that it is a generic skill. Research shows that it is possible to design a curriculum to teach CT. This requires a structured curriculum, with well-defined content, instruction and activities, and suitable formative and summative assessment.[footnote 80]

Based on the above, high-quality computing education may have the following features

- The curriculum is rich in computer science knowledge, enabling pupils to make sense of the entire computing curriculum.
- Pupils learn important programming knowledge to enable them to become skilful programmers.
- The curriculum sets out the knowledge pupils need to build a mental model of program execution.
- Programming languages are chosen to meet curriculum goals.
- Development of CT and problem-solving is underpinned by domain-specific knowledge that is identified and sequenced in the curriculum.

Information technology

Information technology provides a context for the use of computers in society. It focuses on how computers are used in different sectors and describes the methods used to create digital artefacts such as presentations, spreadsheets and videos. In this review, we consider 2 content areas of information technology: digital artefacts and computing contexts. We have chosen these areas because they appear in computing programmes of study,[footnote 81] either explicitly in the case of digital artefacts or implicitly for computing contexts.

Digital artefacts

Digital artefacts are digital objects created by humans. They can be created in a range of media, including text, image, video and sound. It is important that pupils learn the knowledge they need to be confident in using applications in creative projects, including applications that analyse data or manipulate digital artefacts. Declarative and procedural knowledge underpin pupils' ability to create digital artefacts using these applications.[footnote 82] Examples of this knowledge might include:

- commonly used formulae in spreadsheet software, and common methods of manipulating data, such as sorting, filtering and charting
- in image editing: knowledge of bitmap/vector images, layers and colour blending, how to use masking, and how to edit and merge layers and apply filters
- design principles, such as the 'rule of thirds', simplicity, and the use of white space and design patterns

When developing digital artefacts, it is important that pupils can make judgements on trustworthiness, to design products using recognised design elements and to design with usability in mind.[footnote 83] The ability to do this is underpinned by specific subject knowledge. Pupils should have sufficient curriculum time to learn this knowledge, and to have repeated encounters with it to secure it across differing media, products and sources.

Computing contexts

Knowledge of how computing is used purposefully is 'empowering knowledge'.[footnote 84] It sets out the transformative rationale for the subject and the profound impact it has had on humanity. Knowledge of computing contexts chronicles the history of the discipline and explains how computing is used in the modern world. Pupils may learn about the early use of computers such as Colossus, which contributed to saving lives in the Second World War, and technologies that have transformed our lives, such as the internet and the range of services that use it. Knowledge of computing contexts also includes emerging technologies and associated fields, such as data science and artificial intelligence, which are set to shape our future.

The computing programmes of study in the national curriculum state that pupils should learn about common uses of technology beyond school, and about the internet and the World Wide Web, and the opportunities they offer.[footnote 85] These are computing contexts. We know from our review of previous research how knowledge is stored in long-term memory through the building of a range of schemata or complex structures that link knowledge and create meaning.[footnote 86] In applying this general principle to computing contexts, pupils build knowledge in this area by being taught about different contexts over time (breadth) but also by revisiting these contexts and adding new knowledge to what they already know about them (depth). In addition to learning about the contexts themselves, pupils should learn the knowledge that links them together. This includes knowledge of the technologies that enable such contexts, the laws that constrain their use and the ethical considerations when technology intersects with society.[footnote 87]

For example, digital mapping is a common use of technology that pupils might learn about. In primary schools, it might be appropriate to teach pupils about the use of digital mapping and how they can drop a pin anywhere on a map to visit that location virtually. Pupils develop knowledge of how this computing context is useful, such as being able to view a location before visiting it. In secondary schools, this context may be revisited to add to pupils' existing schema by teaching them about the technology that makes digital mapping possible: photos taken by camera arrays on vehicles are stitched together automatically by computer algorithms to build a 3D representation. Pupils may also learn about digital privacy and why faces and other sensitive details are often blurred on these services. As pupils continue their study of computing, they might be taught about the ethical considerations relating to such services and why they are not permitted in certain locations or countries, further adding to their schema.

Based on the above, high-quality computing education may have the following features

- The curriculum to teach pupils how to create digital artefacts is underpinned by specified declarative and procedural knowledge.
- Pupils' schemata of computing contexts is built through new and repeated encounters with contexts to build a breadth and depth of knowledge.

Digital literacy

The National Centre for Computing Education defines digital literacy as the 'skills and knowledge required to be an effective, safe and discerning user of a range of computer systems'.[footnote 88] It covers a range of knowledge and skills, such as using physical devices or knowledge of the features that are likely to mean digital content is reliable.

One of the barriers to pupils developing knowledge in digital literacy is the belief that they are 'digital natives' and already experts in the use of digital devices. Schools might assume that, because pupils have grown up in a world of technology, they do not need to be taught how to use it.[footnote 89] Research indicates that teachers should not make this assumption and that it is important to look at a range of indicators to determine the levels of pupils' knowledge in this area.[footnote 90] For instance, there is evidence that many pupils do not fully consider the reliability of a source and will usually consider it in terms of relevance and personal interest.[footnote 91] In a review of distance learning, the Chartered College of Teaching highlighted that 'despite often being regular users of technology, [pupils] may lack basic digital literacy skills'.[footnote 92] Others go further and argue that the digital native is a fallacy and that a curriculum based on this belief may make it difficult for pupils to acquire new knowledge.[footnote 93] For pupils to use computing devices effectively, they need to be taught how to use them.[footnote 94]

It is important that schools establish a carefully sequenced curriculum for e-safety that builds on what pupils have already learned and identifies subject content that is appropriate for their stage of development. In 2010, we found that, for e-safety: 'There was a close relationship between the provision that the schools made and the pupils' knowledge and understanding'.[footnote 95] It is not enough to set out in the school's policy what pupils should know and remember. This should be rooted in the design of the curriculum and taught by teachers who have had opportunities to develop subject knowledge in online safety.

Based on the above, high-quality computing education may have the following features

- Teachers should not make assumptions about pupils' prior knowledge within digital literacy.
- Knowledge and skills are clearly identified to teach pupils how to use computing devices.
- The curriculum carefully sequences knowledge related to e-safety to ensure that subject content is appropriate for pupils at each stage of their education.

Curriculum sequencing

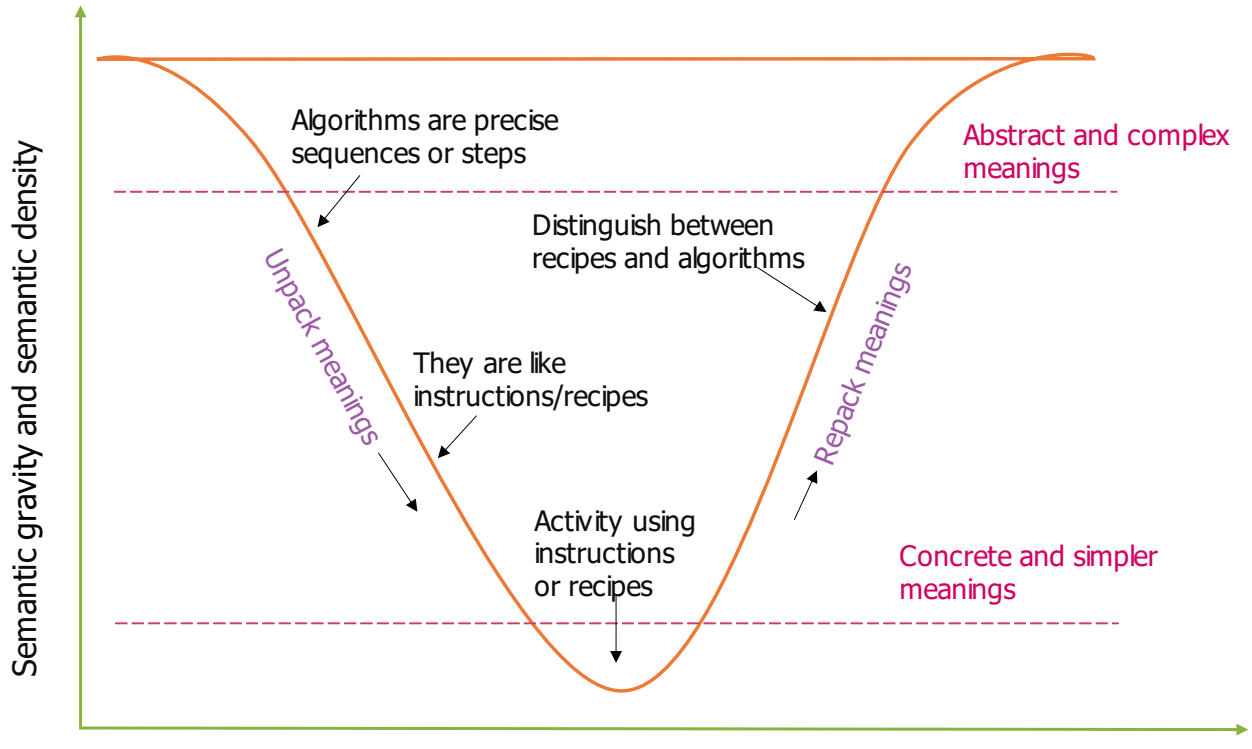
In the previous sections, this review classified declarative and procedural knowledge across key content areas of computing that pupils will learn throughout their time at school. Winch argued that curriculum design should consider how to develop expertise within the subject.[footnote 96] Webb and others build on this argument in a computing context and highlight 'a need for a structured approach to progression in learning the basic facts and central concepts of the subject' in curriculum design.[footnote 97] These views align with the principles underpinning the EIF that the curriculum is the progression model and that subject curriculums should focus on the progression through the content to be learned.

Hubwieser notes that it is important to consider how knowledge components or elements are sequenced when designing a curriculum and that 'knowledge determines the substantial and logical structure of the teaching process'. He acknowledges that it is not possible to teach all computing knowledge and that it is important to select the knowledge that is most important.[footnote 98] Ashbee states that 'a great deal of the knowledge in computer science is procedural and composite, and this characteristic has implications for the structuring of the curriculum'. For example, a pupil's ability to evaluate the reliability of content and the consequences of unreliable content is a composite. To do this well requires sufficient pre-requisite knowledge. This might include knowledge of the types of content, features of reliability, suitable examples of reliable and unreliable content and the potential consequences. This knowledge will need to be carefully sequenced to ensure that it is learned before pupils can be successful in this composite.

Computing is rich in abstract concepts that can be difficult for novices to learn. There is a growing body of research into using 'semantic waves' to sequence subject curriculums and teaching, to enable novices to develop expert knowledge over time.[footnote 99] A semantic wave might start by introducing an abstract concept. The teacher then unpacks the meaning of the concept by putting it into a different context, using simpler, more concrete examples and pedagogical strategies such as metaphors, unplugged activities and worked examples. This knowledge is then repacked and linked back to the original abstract meaning.

Much of the work to explore the use of semantic waves in computing has focused on sequencing across a single lesson.[footnote 100] Therefore, semantic waves could be used to design a curriculum over different timescales, from a single unit of work to a programme of study lasting months or years. An example of how this would apply to the concept of algorithms is shown in Figure 1. Algorithms are introduced at an abstract level as a sequence of steps. The meaning of this is unpacked by comparing an algorithm to a recipe. Activities such as designing recipes are used to teach this point in a concrete way. Teaching then makes a distinction between recipes and algorithms, repacking knowledge back into the abstract form.[footnote 101]

Figure 1: Semantic wave structure[footnote 102]



Source: 'Using semantic waves to improve explanations and learning activities in computing', Teach Computing, February 2020. This source contains public sector information that is licensed under the Open Government Licence v3.0.

Cross-curricular computing

There is some discussion and debate about integrating computing into other curriculum subjects.[footnote 103] Some believe that elements of the computing curriculum, such as those relating to information technology and digital literacy, can be taught through other subjects.[footnote 104] Fluck and others note that integrating computing across the curriculum could be difficult to argue against 'if teachers are knowledgeable, confident, fully trained in computer science and have adequate resources'.[footnote 105] However, we know from what has been explored earlier in this review that this is unlikely to be the case in most schools. In examining approaches to implementing the curriculum, Fluck and others also note how cross-curricular integration of similar subjects, such as ICT, was ineffective and put these subjects in a vague place in the curriculum. Our own research in 2011 highlighted weaknesses in taking a cross-curricular approach to teaching ICT.[footnote 106]:

Recent reviews of international trends found that schools take different approaches to implementing a computing curriculum.[footnote 107] These range from teaching computing as an independent subject to more cross-curricular approaches. One review identified that these curriculum choices are often made in response to a lack of curriculum time or specialist teachers.[footnote 108] This suggests that decisions about curriculum design and implementation are sometimes, understandably, being made to mitigate the effects of these factors rather than because it is the most appropriate way for pupils to learn computing.

In secondary schools, computing is typically taught as a discrete subject; however, this is not always the case.[footnote 109] In 2012, the Royal Society noted that the concept-driven nature of the subject makes it more suited to a discrete curriculum than a cross-curricular approach.[footnote 110] Additionally, a German study from 2010 that compared the effectiveness of different ways of teaching computing suggests that pupils who were taught a discrete computing curriculum tended to perform better than those who were taught through an integrated or cross-curricular approach.[footnote 111]

Based on the above, high-quality computing education may have the following features

- Facts and essential concepts are sequenced to enable pupils to develop expertise within the subject.
- Component declarative and procedural knowledge are identified and sequenced to enable pupils to be successful in learning complex ideas or processes.
- Decisions to teach the subject in a discrete or cross-curricular way are based on how best to teach the intended curriculum.

Pedagogy

Computing is rich in complex knowledge. This can make it interesting for pupils to learn. However, it also requires teachers to consider carefully how to teach the subject content so that all pupils learn this important knowledge. Our previous review of research highlighted the need to manage the cognitive load placed on pupils' short-term memory.[footnote 112]

In applying this to computing, it is important to consider that elements of the subject content have an intrinsically high cognitive load.[footnote 113] That is to say, the subject content places great demands on a pupil's short-term memory. In this section, we explore ways to manage this cognitive load and help pupils remember what is taught.

Novice to expert

Computing has a history of incorporating self-directed modes of learning.[footnote 114] However, some researchers have highlighted that these approaches do not benefit novices and that explicit instructional guidance is important for pupils who lack sufficiently high prior knowledge.[footnote 115]

Robins and others discuss the implications of this for computing education and note that unguided approaches are not appropriate for novices.[footnote 116] It is therefore important that teachers consider pupils' level of expertise when deciding on teaching approaches. Winch describes experts within the subject as those who have sufficient knowledge to acquire knowledge and to challenge and reassess claims within the subject.[footnote 117] Reflecting on this point, Webb claims that this level of computing expertise 'is not accessible to school students but comes in more advanced studies beyond school'.[footnote 118]

Those who promote explicit instructional guidance acknowledge that 'small group and independent problems and projects can be effective – not as vehicles for making discoveries, but as a means of practising recently learned content and skills'.[footnote 119] However, such activities should be used to develop expertise and not be seen as simplified versions of expert practice.[footnote 120]

Worked examples

Computing is a subject full of problems for pupils to solve. These problems can, however, be complex and difficult for those who are new to the subject. As noted

above, instructional guidance is important for pupils who are novices, and these pupils require ‘scaffolding’ to help them develop knowledge.[footnote 121]

One approach to ‘scaffolding’ is using worked examples. Tuovinen and Sweller found that students substantially benefited from worked examples in learning to use a database program.[footnote 122] In this instance, worked examples made less difference for those with more prior knowledge. Skudder and Luxton-Reilly have suggested that using worked examples has positive effects that should transfer to other computing contexts, such as programming problems. They also highlight the benefits of worked examples to novices and their reduced effectiveness as pupils develop expertise.[footnote 123] This difference in effectiveness of worked examples between novices and experts has implications for curriculum sequencing. The use of faded worked examples, where steps are removed from worked examples over time, can be sequenced in the curriculum as pupils develop expertise.[footnote 124]

The use of worked examples can be supported by using labelled subgoals. Subgoals are useful steps in solving complex problems and act as functional components of the problem’s solution.[footnote 125] Labels are added to each subgoal to explain its purpose. This scaffold helps novices to tackle the component parts of a much larger problem. Subgoals can also help pupils see past the surface-level features in problems and transfer this knowledge to new problems. This is because the subgoals focus on abstract knowledge and not the surface features of the problem.[footnote 126]

Margulieux and others discovered that novice students in undergraduate classes who used subgoals in a block-based programming environment were able to complete more steps, more quickly, than novice students who were not given subgoals.[footnote 127] This group were also better able to recall content subsequently. This work highlighted the benefits of subgoals in reducing cognitive load. Robins and others note a further study that found that pupils performed better in text-based programming languages when using subgoals.[footnote 128] More recent findings from schools have highlighted that subgoals are less useful for pupils with substantial prior knowledge.[footnote 129]

Unplugged activities

Unplugged activities take an ‘approach of exposing children to the ideas of computer science without using computers’.[footnote 130] This is particularly beneficial where there is limited access to computing devices. A well-known example of an unplugged activity is for pupils to create a set of instructions for a robot to make a sandwich, which are then carried out by a teacher.[footnote 131]

Research from 2017 found that unplugged activities were one of the most common pedagogical approaches taken by computing teachers in the UK.[footnote 132] Unplugged activities can be useful in introducing computing concepts to pupils at the early stages of their computing education.[footnote 133] However, when choosing unplugged activities, teachers should pay careful attention to how the activity will contribute to achieving curriculum goals. An analysis from 2012 identified that not all unplugged activities are strongly linked to computing subject content.[footnote 134] Furthermore, they might introduce misconceptions. For example, unplugged activities to teach pupils about algorithms can be ambiguous and in opposition to the unambiguous nature of algorithms as precisely defined procedures.[footnote 135]

Teachers should also consider pupils’ prior knowledge when selecting unplugged activities, as these activities appear to be more useful when introducing new knowledge. For instance, when pupils already possess knowledge related to the activity, they can become uninterested in unplugged activities.[footnote 136] Research also indicates that the use of unplugged activities may reduce pupils’ desire to study

computing when these are not well linked to computing concepts.[footnote 137]

Storytelling

Storytelling is one of the oldest methods of sharing and communicating knowledge. Psychologists refer to stories as ‘psychologically privileged’.[footnote 138] This means that they are organised differently in memory than other types of content. The story of the ‘Cat in the hat!’ by Dr Seuss, for example, is a memorable way of helping pupils understand the computing concept of recursion, with increasingly smaller cats appearing to clean a bath tub.[footnote 139]

The structure of stories is important. Willingham highlights that curriculum content can be organised within the lesson using the structure of stories, for example, using causality, conflict, complications and character as a framework for sequencing lessons.[footnote 140] Curzon and others highlight that in computing ‘the links from the story to the technical concepts, again, have to be made clear – travelling the semantic wave’.[footnote 141] This means that, if stories or analogies are used, they should be linked back to the specific computing concept they represent. The teacher should make clear the differences and similarities between the representation and the concept.

Textbooks

A policy paper from 2014 identified the importance of textbooks in high-performing education systems.[footnote 142] This suggested that ‘a supply of high-quality textbooks may provide considerable support to both teachers and pupils’, especially when there is a gap in teachers’ subject knowledge. This is particularly relevant to computing, where we have already noted the issue of teachers’ subject knowledge.[footnote 143] A 2020 short discussion document commissioned by the Chartered Institute for IT (BCS) School Curriculum and Assessment Committee reported that key stakeholders felt that textbooks could be an important resource for teaching computing and that textbooks can provide a progression model through the hierarchical structure of the subject.[footnote 144]

Based on the above, high-quality computing education may have the following features

- Teachers consider pupils’ expertise and prior knowledge when selecting teaching approaches, with novices requiring more explicit instruction.
- The choice of teaching activities is strongly linked to the intended subject content and helps achieve curriculum goals.
- Teachers use worked examples appropriately to help pupils solve problems.
- Textbooks are used as a resource to support teaching in computing.

Assessment

Assessment in computing needs to determine whether pupils remember what they are taught and can apply that knowledge as intended. When assessment drifts

towards checking generic competency-based outcomes, this can mean a loss of focus on checking whether pupils have learned the component knowledge necessary to understand something more complex or perform a more complex task.

Much of the research on assessment in computing focuses primarily on assessing programming knowledge and CT.[footnote 145] There is an 'urgent need' for a better understanding of formative assessment in computing that focuses on the building blocks of knowledge or assessment that targets misconceptions.[footnote 146] Assessment and testing can have negative connotations;[footnote 147] however, research has shown that regular testing can have a positive effect on learning.[footnote 148] It can encourage pupils to study more, reduce their anxiety about tests and increase their engagement with subject content.

Multiple-choice questions can be an effective tool in formative assessment to check pupils' understanding. If quick formative assessments such as multiple-choice questions are well designed, they can help pupils remember important subject content.[footnote 149] However, writing questions with plausible incorrect answers is difficult.[footnote 150] There has been interest in the use of multiple-choice questions in computing.[footnote 151] Lau describes a range of methods for using these in the computing classroom, for example using cards, hands or applications.[footnote 152] Computing classrooms typically have digital devices available that can make it easier to deliver and analyse multiple-choice questions using online platforms. This also allows both teachers and pupils to get immediate feedback to responses.[footnote 153] A project has been established to 'crowd source' high-quality multiple-choice questions for computing.[footnote 154]

Roediger and Karpicke have highlighted that the use of multiple-choice questions can have negative effects on learning.[footnote 155] Incorrect answers can distract pupils and reinforce misconceptions through the effects of retrieval on memory. These negative effects can be eliminated by giving prompt feedback after asking the question. Feedback can be prepared ahead of time, with follow-up questions prepared as necessary.[footnote 156]

There have been calls for the development of concept inventories (CI) for aspects of computing subject content.[footnote 157] CI are an assessment tool designed to determine whether pupils have remembered important component knowledge. This differs from other types of assessments, such as exams, which focus on performing composite tasks. Parker and others encapsulate this in their development of CI for undergraduate novice programmers, noting that:[footnote 158]

> " The distinction between **achievement assessment** and **assessment of learning** is important, as achievement is the accumulation of learning up to a certain time and learning is a change in student's knowledge over time."

CI have had a positive impact on learning in subjects such as science. Although there has been limited research into the use of CI for computing education in schools, there is some research that describes how to apply them.[footnote 159] It highlights the importance of focusing assessment on component knowledge from the intended curriculum.

There has been research into the use of automated tools for assessing computing. Much of this work relates to the automatic assessment of programming knowledge.[footnote 160] 'Parsons problems' are programming problems that focus on small fragments of program code to enable pupils to learn syntactic constructs while maintaining high levels of engagement.[footnote 161] They can be useful for targeting specific knowledge in component programming and can help to identify misconceptions. They are also an alternative to open-ended code-writing questions.[footnote 162] Parsons problems are solved by arranging blocks of code in the correct order. Focusing on small fragments of code reduces the scope of logic and the pupil's cognitive load.[footnote 163] The use of digital platforms to design and use Parsons problems also allows for pupils to receive immediate feedback.

Research has highlighted that the use of adaptive problems (problems that change based on responses) increases pupils' success in solving problems and reduces frustration.[footnote 164] Distractors (incorrect answers) are a useful element of Parsons problems to help identify misconceptions; however, results from Harms and others found that 'the distractors increased learners' cognitive load, decreased their success at completing Parsons problems and increased learners' time on task.[footnote 165] Teachers should consider carefully how they use distractors in such problems.

Based on the above, high-quality computing education may have the following features

- Assessment focuses on the knowledge and skills identified in the curriculum and not generic competencies.
- Formative assessment is used to identify misconceptions early.

Systems

Teacher subject knowledge

The research underpinning the EIF identified the importance of teachers' subject knowledge.[footnote 166] The Royal Society has highlighted that the levels of CPD carried out by computing teachers are a source of concern. It found 'large disparities in the CPD hours undertaken by computing teachers, with some teachers not receiving any CPD at all'.[footnote 167] This is true in England, where teachers have been asked to teach a computing curriculum with little training.

In a 2017 study, Sentance and Csizmadia found that subject knowledge was the most common challenge facing teachers in teaching a computing curriculum.[footnote 168] Training courses and additional materials can still leave teachers lacking the confidence to help pupils with computing problems. In their discussion of computing curriculum development in Israel and the United States, Gal-Ezer and Stephenson highlighted that teachers' lack of technical, content and pedagogical knowledge is one of the challenges in designing and implementing computing curriculums. In a small-scale study, Qian and others found that teachers who had computer science degrees or who had received additional computing training were better at identifying and addressing students' misconceptions.

Given the importance of teachers' subject knowledge, and the challenges they continue to face in developing this, it is important that school leaders provide sufficient subject-specific professional development to enable teachers to design and teach a high-quality computing curriculum. This is particularly important for computing, where a high proportion of teachers are unlikely to be subject specialists.[footnote 169]

Timetabling/curriculum time

As noted earlier in this report, the amount of curriculum time afforded to computing education is a significant concern within the sector. The Royal Society has found that

the typical time given to computing in the curriculum is insufficient to teach the computing subject content in the national curriculum.[footnote 170] This, combined with the trend for less time for computing in the curriculum, makes it even more difficult for schools to achieve the aims of the national curriculum.[footnote 171] Reducing computing curriculum time also increases the workload of teachers, who must meet the same requirements of the curriculum in less time or provide a diminished curriculum.[footnote 172]

Infrastructure

An ambitious computing curriculum may require teachers to use specialist software and hardware, such as programming environments, microcontrollers, robots or network devices. This requirement can often be at odds with the need to maintain network security and performance in schools. In 2012, the Royal Society highlighted features of school infrastructure as one of the main barriers to effective teaching of computing and likened this issue to the impact of ‘overzealous implementation’ of health and safety rules on school field trips and science experiments.[footnote 173]

In guidance to help those responsible for IT infrastructure, Computing at School highlighted in 2013 that ‘it is possible to teach Computer Science with support from senior leadership, a constructive dialog with the service provider and the cooperation from a student body supported by a well understood Acceptable Use Policy’.[footnote 174] While pupils’ safety should be paramount for school leaders, it is important that perceived risks are weighed up and not used to limit the computing curriculum, unnecessarily denying pupils access to important knowledge and opportunities.

Based on the above, high-quality computing education may have the following features

- Teachers have access to high-quality computing CPD to develop and maintain their subject knowledge.
- Leaders and teachers use the expertise of subject communities to develop teachers’ subject knowledge.
- Adequate curriculum time is allocated to computing.
- Stakeholders work together to ensure that risks are weighed up and do not limit the ambition of the computing curriculum.

Conclusion

This review has explored a range of evidence relating to high-quality computing education. It has drawn on research from many different countries and organisations. It also builds from the same research base that underpins the EIF.

Computing education is important for pupils to make sense of and to contribute positively to our technologically diverse world. This review has highlighted approaches to constructing, sequencing and teaching a coherent computing curriculum rich in computer science, information technology and digital literacy to achieve this aim and the aims set out in the national curriculum. Central to this is the importance of identifying and ordering the underlying knowledge that pupils require to make sense of complex ideas or engage in composite tasks or activities within the subject. Computing is rich in these ideas and tasks, so this is essential. To ensure

that pupils can make progress through the curriculum, it is important that teachers check this knowledge so that pupils are ready for what comes next.

Computing lessons can place great demands on pupils' working memory. Teaching must work to manage this demand and ensure that pupils can think about the intended subject content. Due to the hierarchical nature of many aspects of computing subject knowledge, it is important that pupils' prior knowledge is taken into account when planning teaching and in the selection of teaching activities.

In this review, we have focused on the number of specialist staff in schools. The number of subject specialists in computing is low, and there is a lack of new teachers to improve the situation. This will have significant consequences for the quality of education that pupils receive in computing if nothing is done to remedy the situation. This further strengthens the argument for a focus on subject-specific CPD.

---

1. 'Regulation for the Fourth Industrial Revolution', Department for Business, Energy and Industrial Strategy, June 2019; K Schwab, 'The Fourth Industrial Revolution: what it means and how to respond', World Economic Forum, January 2016.↩
2. 'National curriculum in England: framework for key stages 1 to 4', Department for Education, December 2014; 'National curriculum in England: computing programmes of study', Department for Education, September 2013.↩
3. 'Principles behind Ofsted's research reviews and subject reports', Ofsted, March 2021.↩
4. 'Education inspection framework: overview of research', Ofsted, July 2019.↩
5. I Livingstone and A Hope, 'Next gen: transforming the UK into the world's leading talent hub for the video games and visual effects industries: a review by Ian Livingstone and Alex Hope', Nesta, 2011; S Furber, 'Shut down or restart? The way forward for computing in UK schools', January 2012; LR Larke, 'Agentic neglect: teachers as gatekeepers of England's national computing curriculum', in 'British Journal of Educational Technology', Volume 50, Issue 3, 2019, pages 1,137 to 1,150.↩
6. 'Consultation on (i) the order for replacing ICT with computing and (ii) the regulations for disapplying aspects of the existing national curriculum', Department for Education, May 2013.↩
7. LR Larke, 'Agentic neglect: teachers as gatekeepers of England's national computing curriculum', in 'British Journal of Educational Technology', Volume 50, Issue 3, 2019, pages 1,137 to 1,150, quote on page 1,139.↩
8. A Fluck, M Webb, M Cox, C Angeli, J Malyn-Smith, J Voogt and J Zagami, 'Arguing for computer science in the school curriculum, in 'Educational Technology & Society', Volume 19, Issue 3, 2016, page XIX.↩
9. P Tait, 'After the reboot: computing education in UK schools', The Royal Society, November 2017, page 6.↩
10. 'National curriculum in England: framework for key stages 1 to 4', Department for Education, December 2014.↩
11. 'School inspection handbook', Ofsted, October 2021.↩
12. S Grover, R Pea and S Cooper, 'Designing for deeper learning in a blended computer science course for middle school students', in 'Computer Science Education', Volume 25, Issue 2, 2015, pages 199 to 237, quote on page 200.↩
13. 'Statutory framework for the early years foundation stage', Department for Education, September 2021.↩
14. A Manches and L Plowman, 'Computing education in children's early years: a call for debate', in 'British Journal of Educational Technology', Volume 48, Issue 1, 2017, pages 191 to 201.↩

15. MU Bers, C González-González and MB Armas-Torres, ‘Coding as a playground: promoting positive learning experiences in childhood classrooms’, in ‘Computers and Education’, Volume 138, 2019, pages 130 to 145; MU Bers, L Flannery, ER Kazakoff and A Sullivan, ‘Computational thinking and tinkering: exploration of an early childhood robotics curriculum’, in ‘Computers and Education’, Volume 72, 2014, pages 145 to 157; MU Bers, ‘The TangibleK robotics program: applied computational thinking for young children’, in ‘Early Childhood Research and Practice’, Volume 12, Issue 2, 2010.↩
16. A Manches and L Plowman, ‘Computing education in children’s early years: a call for debate’, in ‘British Journal of Educational Technology’, Volume 48, Issue 1, 2017, pages 191 to 201.↩
17. ‘National curriculum in England: computing programmes of study’, Department for Education, September 2013.↩
18. P Tait, ‘After the reboot: computing education in UK schools’, The Royal Society, November 2017.↩
19. P Tait, ‘After the reboot: computing education in UK schools’, The Royal Society, November 2017, page 36.↩
20. PJ Rich, SF Browning, MK Perkins, T Shoop, E Yoshikawa and OM Belikov, ‘Coding in K-8: international trends in teaching elementary/primary computing’, in ‘TechTrends’, Volume 63, Issue 3, 2019, pages 311 to 329.↩
21. ‘National curriculum in England: computing programmes of study’, Department for Education, September 2013.↩
22. P Tait, ‘After the reboot: computing education in UK schools’, The Royal Society, November 2017.↩
23. PEJ Kemp and M Berry, ‘The Roehampton annual computing education report: pre-release snapshot from 2018’, University of Roehampton, May 2019.↩
24. P Tait, ‘After the reboot: computing education in UK schools’, The Royal Society, November 2017.↩
25. PEJ Kemp and M Berry, ‘The Roehampton annual computing education report: pre-release snapshot from 2018’, University of Roehampton, May 2019.↩
26. [‘National trends in A-level, AS-level and GCSE entries and grades’, FFT education datalab.↩
27. ‘National trends in A-level, AS-level and GCSE entries and grades’, FFT education datalab.↩
28. ‘ICT in schools: 2008 to 2011’, Ofsted, December 2011.↩
29. ‘National trends in A-level, AS-level and GCSE entries and grades’, FFT Education Datalab.↩
30. P Tait, ‘After the reboot: computing education in UK schools’, The Royal Society, November 2017, page 38.↩
31. ‘CAS: females and computing/computer science survey (2018)’, Computing at School, May 2018.↩
32. ‘Young people’s views on science education: science education tracker 2019, wave 2’, Wellcome Trust, March 2020.↩
33. M Kallia and S Sentance, ‘Are boys more confident than girls? The role of calibration and students’ self-efficacy in programming tasks and computer science’, in ‘Proceedings of the 13th Workshop in Primary and Secondary Computing Education: WIPSCE ‘18’, Association for Computing Machinery, 2018.↩
34. PEJ Kemp, B Wong and MG Berry, ‘Female performance and participation in computer science’, in ‘ACM Transactions on Computing Education’, Volume 20, Issue 1, 2020, pages 1 to 28. The other subjects were French, German and Spanish. Since the research was carried out, adjustments have been made to the

grading standards in French and German to make grading less severe. ‘Grading standards in GCSE French, German and Spanish’, Ofqual, November 2019.↩

35. LS Shulman, ‘Those who understand: knowledge growth teaching’, in ‘Educational Researcher’, Volume 15, Issue 2, 1986, pages 4 to 14.↩
36. J Worth and J Van Den Brande, ‘Retaining science, mathematics and computing teachers’, National Foundation for Educational Research, November 2019.↩
37. P Tait, ‘After the reboot: computing education in UK schools’, The Royal Society, November 2017.↩
38. P Tait, ‘After the reboot: computing education in UK schools’, The Royal Society, November 2017.↩
39. ‘Reporting year 2019: school workforce in England’, National Statistics, June 2020.↩
40. J Worth and J Van Den Brande, ‘Retaining science, mathematics and computing teachers’, National Foundation for Educational Research, November 2019.↩
41. ‘Academic year 2020/21: initial teacher training census’, Department for Education, December 2020; ‘School workforce in England; November 2020’, Department for Education, June 2021.↩
42. ‘Initial teacher training (ITT) census for 2020 to 2021’, Department for Education, December 2020.↩
43. ‘Academic year 2020/21: initial teacher training census’, Department for Education, December 2020.↩
44. ‘Policy briefing on teachers of computing: recruitment, retention and development’, The Royal Society, 2019.↩
45. ‘Education inspection framework: overview of research’, Ofsted, January 2019.↩
46. P Tait, ‘After the reboot: computing education in UK schools’, The Royal Society, November 2017; ‘National curriculum in England: computing programmes of study’, Department for Education, September 2013.↩
47. ‘Education inspection framework (EIF)’, Ofsted, July 2021.↩
48. W Lau, ‘Teaching computing in secondary schools: a practical handbook’, Routledge, 2017.↩
49. AV Robins, LE Margulieux and BB Morrison, ‘Cognitive sciences for computing education’, in ‘The Cambridge handbook of computing education research’, edited by S Fincher and A Robins, Cambridge University Press, 2019, pages 231 to 275.↩
50. K Lawless and J Kulikowich, ‘Domain knowledge and individual interest: the effects of academic level and specialization in statistics and psychology’, in ‘Contemporary Educational Psychology’, Volume 31, 2006, pages 30 to 43.↩
51. ‘National curriculum in England: computing programmes of study’, Department for Education, September 2013; M Webb, N Davis, T Bell, Y Katz, N Reynolds, DP Chambers and MM Sysło, ‘Computer science in K-12 school curricula of the 21st century: why, what and when?’, in ‘Education and Information Technologies’, Volume 22, Issue 2, 2017, pages 445 to 468.↩
52. S Peyton Jones, ‘Why we should teach children to code’, 2020.↩
53. S Sentance, J Waite and M Kallia, ‘Teaching computer programming with PRIMM: a sociocultural perspective’, in ‘Computer Science Education’, Volume 29, Issues 2 to 3, 2019, pages 136 to 176; S Grover, R Pea and S Cooper, ‘Designing for deeper learning in a blended computer science course for middle school students’, in ‘Computer Science Education’, Volume 25, Issue 2, 2015, pages 199 to 237; B Du Boulay, ‘Some difficulties of learning to program’, in ‘Journal of Educational Computing Research’, Volume 2, Issue 1, 1986, pages 57 to 73; S Sentance and A Csizmadia, ‘Computing in the curriculum: challenges and strategies from a teacher’s perspective’, in ‘Education and Information

Technologies', Volume 22, Issue 2, 2017, pages 469 to 495.↩

54. C Schulte, 'Block model: an educational model of program comprehension as a tool for a scholarly approach to teaching', ICER'08 – Proceedings of the ACM Workshop on International Computing Education Research, 2008.↩

55. C Schulte, T Clear, A Taherkhani, T Busjahn and JH Paterson, 'An introduction to program comprehension for computer science educators', in 'Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE', 2010, quotes on page 83.↩

56. 'Education inspection framework: overview of research', Ofsted, January 2019.↩

57. AV Robins, LE Margulieux and BB Morrison, 'Cognitive sciences for computing education', in 'The Cambridge handbook of computing education research', edited by S Fincher and A Robins, Cambridge University Press, 2019, page 242.↩

58. B Du Boulay, 'Some difficulties of learning to program', in 'Journal of Educational Computing Research', Volume 2, Issue 1, 1986, pages 57 to 73.↩

59. J Sorva, 'Misconceptions and the beginner programmer', in 'Computer science education: perspectives on teaching and learning in school', edited by S Sentance, E Barendsen and C Schulte, Bloomsbury Publishing, 2018, pages 171 to 187.↩

60. 'National curriculum in England: computing programmes of study', Department for Education, September 2013.↩

61. J Sorva, 'Misconceptions and the beginner programmer', in 'Computer science education: perspectives on teaching and learning in school', edited by S Sentance, E Barendsen, and C Schulte, Bloomsbury Publishing, 2018, pages 171 to 187.↩

62. 'National curriculum in England: computing programmes of study', Department for Education, September 2013.↩

63. P Tait, 'After the reboot: computing education in UK schools', The Royal Society, November 2017.↩

64. O Meerbaum-Salant, M Armoni and M Ben-Ari, 'Habits of programming in Scratch', ITiCSE '11: Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education, 2011, quotes on pages 171 and 172.↩

65. S Grover and S Basu, 'Measuring student learning in introductory block-based programming: examining misconceptions of loops, variables, and Boolean logic', in 'Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education', Association for Computing Machinery, 2017.↩

66. K Falkner, S Sentance, R Vivian, S Barksdale, L Busuttil, and others, 'An international comparison of K-12 computer science education intended and enacted curricula', in 'ACM International Conference Proceeding Series', Association for Computing Machinery, 2019, quote on page 9.↩

67. L Grandell, M Peltomäki, R-J Back and T Salakoski, 'Why complicate things? introducing programming in high school using python', in 'Proceedings of the 8th Australasian Conference on Computing Education – Volume 52 January 2006 Pages 71 to 80', 2006;↩

68. 'Education inspection framework (EIF)', Ofsted, 2021.↩

69. 'National curriculum in England: computing programmes of study', Department for Education, September 2013.↩

70. S Papert, 'Mindstorms: children, computers and powerful ideas', Basic Books, Inc, 1980, page 182.↩

71. JM Wing, 'Computational thinking', in 'Communications of the ACM', Volume 49, Issue 3, 2006, pages 33 to 35, quote on page 33.↩

72. M Tedre and PJ Denning, 'The long quest for computational thinking', in 'ACM International Conference Proceeding Series', Association for Computing Machinery, 2016.↩

73. C Selby, M Dorling and J Woollard, ‘Evidence of assessing computational thinking’, December 2014.↩

74. V Barr and C Stephenson, ‘Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community?’, in ‘ACM Inroads’, Volume 2, Issue 1, 2011, pages 48 to 54.↩

75. S Grover and R Pea, ‘Computational thinking in K-12: a review of the state of the field’, in ‘Educational Researcher’, Volume 42, Issue 1, 2013, pages 38 to 43.↩

76. S Grover and RD Pea, ‘Computational thinking: a competency whose time has come’, in ‘Computer science education: perspectives on teaching and learning in school’, edited by S Sentance, E Barendsen and C Schulte, Bloomsbury Academic, 2018.↩

77. A Tricot and J Sweller, ‘Domain-specific knowledge and why teaching generic skills does not work’, in ‘Educational Psychology Review’, Volume 26, Issue 2, 2014, pages 265 to 283.↩

78. T Crick, ‘Computing education: an overview of research in the field’, April 2017, pages 1 to 38.↩

79. T Crick, ‘Computing education: an overview of research in the field’, Issue April, 2017, pages 1 to 38.↩

80. S Grover, R Pea and S Cooper, ‘Designing for deeper learning in a blended computer science course for middle school students’, in ‘Computer Science Education’, Volume 25, Issue 2, 2015, page 3.↩

81. ‘National curriculum in England: computing programmes of study’, Department for Education, September 2013.↩

82. E Barendsen, N Grgurina and J Tolboom, ‘A new informatics curriculum for secondary education in the Netherlands’, in ‘Informatics in schools: improvement of informatics knowledge and perception’, ISSEP, Springer, 2016.↩

83. ‘National curriculum in England: computing programmes of study’, Department for Education, September 2013.↩

84. R Ashbee, ‘Curriculum: theory, culture and the subject specialisms’, Routledge, 2021.↩

85. ‘National curriculum in England: computing programmes of study’, Department for Education, September 2013.↩

86. ‘Education inspection framework: overview of research’, Ofsted, January 2019.↩

87. R Ashbee, ‘Curriculum: theory, culture and the subject specialisms’, Routledge, 2021.↩

88. ‘Digital literacy within the computing curriculum’, Teach Computing, January 2021.↩

89. ‘The fallacy of the “digital native”’, International Computer Driving Licence, 2014.↩

90. EJ Helsper and R Eynon, ‘Digital natives: where is the evidence?’, in ‘British Educational Research Journal’, Volume 36, Issue 3, 2010, pages 503 to 520.↩

91. T Lumley and J Mendelovits, ‘How well do young people deal with contradictory and unreliable information online? What the PISA digital reading assessment tells us’, paper presented at the the Annual Conference of the American Educational Research Association, April 2012.↩

92. LM Muller and G Goldenberg, ‘Education in times of crisis: effective approaches to distance learning’, Chartered College of Teaching, November 2021, quote on page 8.↩

93. PA Kirschner and P De Bruyckere, ‘The myths of the digital native and the multitasker’, in ‘Teaching and Teacher Education’, Volume 67, 2017, pages 135 to 142.↩

94. M Jakubowski, ‘Computers at schools: it’s not enough to have them and it’s not enough to use them’, December 2014.↩

95. ‘The safe use of new technologies’, Ofsted, February 2010, quote on page 4.↩
96. C Winch, ‘Curriculum design and epistemic ascent’, in ‘Journal of Philosophy of Education’, Volume 47, Issue 1, 2013, pages 128 to 146.↩
97. M Webb, N Davis, T Bell, Y Katz, N Reynolds, DP Chambers and MM Sysło, ‘Computer science in K-12 school curricula of the 21st century: why, what and when?’, in ‘Education and Information Technologies’, Volume 22, Issue 2, 2017, pages 445 to 468.↩
98. P Hubwieser, ‘Computer science education in secondary schools – the introduction of a new compulsory subject’, in ‘ACM Transactions on Computing Education’, Volume 12, Issue 4, 2012.↩
99. K Maton, ‘Making semantic waves: a key to cumulative knowledge-building’, in ‘Linguistics and Education’, Volume 24, Issue 1, 2013, pages 8 to 22.↩
100. R Ashbee, ‘Curriculum: theory, culture and the subject specialisms’, Routledge, 2021.↩
101. ‘Using semantic waves to improve explanations and learning activities in computing’, Teach Computing, February 2020.↩
102. Semantic gravity is defined as how context-dependent a concept is. Semantic density defines concept complexity.↩
103. A Fluck, M Webb, M Cox, C Angeli, J Malyn-Smith, J Voogt and J Zagami, ‘Arguing for computer science in the school curriculum, in ‘Educational Technology & Society’, Volume 19, Issue 3, 2016, pages 38 to 46; M Webb, N Davis, T Bell, Y Katz, N Reynolds, DP Chambers and MM Sysło, ‘Computer science in K-12 school curricula of the 21st century: why, what and when?’, in ‘Education and Information Technologies’, Volume 22, Issue 2, 2017, pages 445 to 468; ME Webb, T Bell, N Davis, YJ Katz, A Fluck, and others, ‘Tensions in specifying computing curricula for K-12: towards a principled approach for objectives’, in ‘it – Information Technology’, Volume 60, Issue 2, 2018, pages 59 to 68; O McGarr and K Johnston, ‘Curricular responses to Computer Science provision in schools: current provision and alternative possibilities’, in ‘Curriculum Journal’, Volume 31, Issue 4, 2020, pages 745 to 756.↩
104. D Wells, ‘Embedding information and communication technology across the curriculum – where are we at?’, in ‘Research in Teacher Education’, Volume 4, Issue 2, 2014, pages 11 to 16; G Polizzi, ‘Digital literacy and the national curriculum for England: learning from how the experts engage with and evaluate online content’, in ‘Computers and Education’, Volume 152, 2020.↩
105. A Fluck, M Webb, M Cox, C Angeli, J Malyn-Smith, J Voogt and J Zagami, ‘Arguing for computer science in the school curriculum, in ‘Educational Technology & Society’, Volume 19, Issue 3, 2016, page XIX.↩
106. ‘ICT in schools: 2008 to 2011’, Ofsted, December 2011.↩
107. PJ Rich, SF Browning, MK Perkins, T Shoop, E Yoshikawa and OM Belikov, ‘Coding in K-8: international trends in teaching elementary/primary computing’, in ‘TechTrends’, Volume 63, Issue 3, 2019, pages 311 to 329; F Heintz, L Mannila and T Färnqvist, ‘A review of models for introducing computational thinking, computer science and computing in K-12 education’, IEEE Frontiers in Education Conference (FIE), 2016, pages 1 to 9.↩
108. F Heintz, L Mannila and T Färnqvist, ‘A review of models for introducing computational thinking, computer science and computing in K-12 education’, IEEE Frontiers in Education Conference (FIE), 2016, pages 1 to 9.↩
109. P Tait, ‘After the reboot: computing education in UK schools’, The Royal Society, November 2017; F Heintz, L Mannila and T Färnqvist, ‘A review of models for introducing computational thinking, computer science and computing in K-12 education’, IEEE Frontiers in Education Conference (FIE), 2016, pages 1 to 9.↩
110. S Furber, ‘Shut down or restart? The way forward for computing in UK schools’, The Royal Society, January 2012.↩

111. C Steer and P Hubwieser, ‘Comparing the efficiency of different approaches to teach informatics at secondary schools’, in ‘Informatics in Education’, 2010, page IX.↩

112. ‘Education inspection framework: overview of research’, Ofsted, January 2019.↩

113. P Hubwieser, ‘Computer science education in secondary schools – the introduction of a new compulsory subject’, in ‘ACM Transactions on Computing Education’, Volume 12, Issue 4, 2012; AV Robins, L Margulieux and BB Morrison, ‘Cognitive sciences for computing education’, in ‘The Cambridge handbook of computing education research’, edited by SA Fincher and AV Robins, Cambridge University Press, 2019, pages 231 to 275.↩

114. E Ackermann, ‘Piaget’s constructivism, Papert’s constructionism: what’s the difference?’, in ‘Future of Learning Group Publication’, Volume 5, Issue 3, 2001, page 438.↩

115. PA Kirschner, J Sweller and RE Clark, ‘Why minimal guidance during instruction does not work: an analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching’, in ‘Educational Psychologist’, Volume 41, Issue 2, 2006, pages 75 to 86.↩

116. AV Robins, L Margulieux and BB Morrison, ‘Cognitive sciences for computing education’, in ‘The Cambridge handbook of computing education research’, edited by SA Fincher and AV Robins, Cambridge University Press, 2019, pages 231 to 275.↩

117. C Winch, ‘Curriculum design and epistemic ascent’, in ‘Journal of Philosophy of Education’, Volume 47, Issue 1, 2013, pages 128 to 146.↩

118. M Webb, N Davis, T Bell, Y Katz, N Reynolds, DP Chambers and MM Sysło, ‘Computer science in K-12 school curricula of the 21st century: why, what and when?’, in ‘Education and Information Technologies’, Volume 22, Issue 2, 2017, pages 445 to 468, quote on page 461.↩

119. RE Clark, PA Kirschner and J Sweller, ‘Putting students on the path to learning: the case for fully guided instruction’, in ‘The American Educator’, Volume 36, 2012, pages 6 to 11, quote on page 6.↩

120. C Winch, ‘Curriculum design and epistemic ascent’, in ‘Journal of Philosophy of Education ‘, Volume 47, Issue 1, 2013, pages 128 to 146.↩

121. PA Kirschner, J Sweller and RE Clark, ‘Why minimal guidance during instruction does not work: an analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching’, in ‘Educational Psychologist’, Volume 41, Issue 2, 2006, pages 75 to 86.↩

122. JE Tuovinen and J Sweller, ‘A comparison of cognitive load associated with discovery learning and worked examples’, in ‘Journal of Educational Psychology’, Volume 91, Issue 2, 1999, pages 334 to 341.↩

123. B Skudder and A Luxton-Reilly, ‘Worked examples in computer science’, in ‘Proceedings of the Sixteenth Australasian Computing Education Conference, Volume 148’, 2014, pages 59 to 64.↩

124. A Renkl, RK Atkinson, UH Maier and R Stanley, ‘From example study to problem solving: smooth transitions help learning’, in ‘The Journal of Experimental Education’, Volume 70, Issue 4, 2002, pages 293 to 315.↩

125. L Margulieux, R Catrambone and M Guzdial, ‘Employing subgoals in computer programming education’, in ‘Computer Science Education’, Volume 26, Issue 1, 2016, pages 44 to 67.↩

126. AV Robins, L Margulieux and BB Morrison, ‘Cognitive sciences for computing education’, in ‘The Cambridge handbook of computing education research’, edited by SA Fincher and AV Robins, Cambridge University Press, 2019, pages 231 to 275.↩

127. LE Margulieux, M Guzdial and R Catrambone, ‘Subgoal-labeled instructional

material improves performance and transfer in learning to develop mobile applications', in 'Proceedings of the Ninth Annual International Conference on International Computing Education Research', ACM, New York, 2012, pages 71 to 78.↩

128. AV Robins, L Margulieux and BB Morrison, 'Cognitive sciences for computing education', in 'The cambridge handbook of computing education research', edited by SA Fincer and AV Robins, Cambridge University Press, 2019, pages 231 to 275.↩

129. LE Margulieux, BB Morrison, B Franke and H Ramilison, 'Effect of implementing subgoals in code.org's intro to programming unit in computer science principles', in 'ACM Transactions on Computing Education', Volume 20, Issue 4, 2020.↩

130. T Bell, J Alexander, I Freeman and M Grimley, 'Computer science unplugged: school students doing real computing without computers', in 'New Zealand Journal of Applied Computing and Information Technology', Volume 13, Issue 1, 2009, pages 20 to 29 quote on page 21.↩

131. 'Unplugged activity: pb & j', Microsoft MakeCode.↩

132. S Sentance and A Csizmadia, 'Computing in the curriculum: challenges and strategies from a teacher's perspective', in 'Education and Information Technologies', Volume 22, Issue 2, 2017, pages 469 to 495.↩

133. WJ Rijke, L Bollen, THS Eysink and JLJ Tolboom, 'Computational thinking in primary school: an examination of abstraction and decomposition in different age groups', in 'Informatics in Education', Volume 17, Issue 1, 2018, pages 77 to 92.↩

134. R Taub, M Armoni and M Ben-Ari, 'CS unplugged and middle-school students' views, attitudes, and intentions regarding CS', in 'ACM Transactions on Computing Education', Volume 12, Issue 2, 2012.↩

135. J Waite, P Curzon, W Marsh and S Sentance, 'Difficulties with design: the challenges of teaching design in K-5 programming', in 'Computers and Education', Volume 150, 2020.↩

136. T Bell and J Vahrenhold, 'CS unplugged — how is it used, and does it work?', in 'Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)', Springer Verlag, 2018, pages 497 to 521.↩

137. R Taub, M Armoni and M Ben-Ari, 'CS unplugged and middle-school students' views, attitudes, and intentions regarding CS', in 'ACM Transactions on Computing Education', Volume 12, Issue 2, 2012.↩

138. DT Willingham, 'Why don't students like school?', second edition, Jossey-Bass, 2021.↩

139. P Curzon, PW McOwan, J Donohue, S Wright and W Marsh, 'Teaching of concepts', in 'Computer science education: perspectives on teaching and learning in school', edited by S Sentance, E Barendsen and C Schulte, Bloomsbury Publishing, 2018, page 94.↩

140. DT Willingham, 'Why don't students like school?', second edition, Jossey-Bass, 2021.↩

141. P Curzon, PW McOwan, J Donohue, S Wright and W Marsh, 'Teaching of concepts', in 'Computer science education: perspectives on teaching and learning in school', edited by S Sentance, E Barendsen and C Schulte, Bloomsbury Publishing, 2018, quote on page 93.↩

142. T Oates, 'Why textbooks count: a policy paper', Cambridge Assessment 2014.↩

143. P Tait, 'After the reboot: computing education in UK schools', The Royal Society, November 2017.↩

144. J Woollard, 'Textbooks for the teaching of computing', 2020.↩

145. M Kallia, 'Assessment in computer science courses: a literature review', The Royal

Society, 2017.↩

146. S Grover, ‘Toward a framework for formative assessment of conceptual learning in K-12 computer science classrooms’, in ‘SIGCSE 2021 – proceedings of the 52nd ACM technical symposium on computer science education’, Association for Computing Machinery, Inc, 2021.↩
147. D Christodoulou, ‘Making good progress? The future of assessment for learning’, Oxford University Press, 2017.↩
148. HL Roediger and JD Karpicke, ‘The power of testing memory: basic research and implications for educational practice’, in ‘Perspectives on Psychological Science’, Volume 1, Issue 3, 2006.↩
149. D Christodoulou, ‘Making good progress? The future of assessment for learning’, Oxford University Press, 2017.↩
150. JL Little, EL Bjork, RA Bjork and G Angello, ‘Multiple-choice tests exonerated, at least of some charges: fostering test-induced learning and avoiding test-induced forgetting’, in ‘Psychological Science’, Volume 23, Issue 11, 2012, pages 1,337 to 1,344.↩
151. J Robinson, ‘Designing and using multiple choice questions in computing’, Teach Computing, October 2019.↩
152. W Lau, ‘Teaching computing in secondary schools: a practical handbook’, Routledge, 2017.↩
153. S Sentance, C Selby and M Kallia, ‘Assessment in the computing classroom’, in ‘Computer science education: perspectives on teaching and learning in school’, edited by S Sentance, E Barendsen and C Schulte, Bloomsbury Academic, 2018, pages 151 to 166.↩
154. ‘A collection of computing quizzes’, Project Quantum.↩
155. HL Roediger and JD Karpicke, ‘The power of testing memory: basic research and implications for educational practice’, in ‘Perspectives on Psychological Science’, Volume 1, Issue 3, 2006.↩
156. D Christodoulou, ‘Making good progress? The future of assessment for learning’, Oxford University Press, 2017.↩
157. A Rachmatullah, B Akram, D Boulden, B Mott, K Boyer, J Lester and E Wiebe, ‘Development and validation of the middle grades computer science concept inventory (mg-csci) assessment’, in ‘Eurasia Journal of Mathematics, Science and Technology Education’, Volume 16, Issue 5, 2020.↩
158. MC Parker, M Guzdial and S Engleman, ‘Replication, validation, and use of a language independent cs1 knowledge assessment’, in ‘ICER 2016 – proceedings of the 2016 ACM conference on international computing education research’, Association for Computing Machinery, Inc, 2016.↩
159. A Rachmatullah, B Akram, D Boulden, B Mott, K Boyer, J Lester and E Wiebe, ‘Development and validation of the middle grades computer science concept inventory (mg-csci) assessment’, in ‘Eurasia Journal of Mathematics, Science and Technology Education’, Volume 16, Issue 5, 2020.↩
160. M Kallia, ‘Assessment in computer science courses: a literature review’, The Royal Society, 2017.↩
161. D Parsons and P Haden, ‘Parson’s programming puzzles: a fun and effective learning tool for first programming courses’, in ‘Computing Education 2006. Proceedings of the Eighth Australasian Computing Education Conference’, Volume 52, edited by D Tolhurst and S Mann, Australian Computing Society, 2006, pages 157 to 163.↩
162. S Sentance and M Kallia, ‘Assessment of computer science’, in ‘Computer science education: perspectives on teaching and learning in school’, edited by S Sentance, E Barendsen and C Schulte, Bloomsbury, 2018.↩

163. AV Robins, L Margulieux and BB Morrison, ‘Cognitive sciences for computing education’, in ‘The Cambridge handbook of computing education research’, edited by SA Fincer and AV Robins, Cambridge University Press, 2019, pages 231 to 275.↩

164. B Ericson, A McCall and K Cunningham, ‘Investigating the affect and effect of adaptive parsons problems’, in ‘9th Koli Calling International Conference on Computing Education Research (Koli Calling ‘19)’, Association for Computing Machinery, 2019.↩

165. KJ Harms, J Chen and C Kelleher, ‘Distractors in Parsons problems decrease learning efficiency for young novice programmers’, in ‘ICER 2016 – proceedings of the 2016 ACM conference on international computing education research’, Association for Computing Machinery, Inc, 2016.↩

166. ‘Education inspection framework: overview of research’, Ofsted, January 2019.↩

167. P Tait, ‘After the reboot: computing education in UK schools’, The Royal Society, November 2017.↩

168. S Sentance and A Csizmadia, ‘Computing in the curriculum: challenges and strategies from a teacher’s perspective’, in ‘Education and Information Technologies’, Volume 22, Issue 2, 2017, pages 469 to 495.↩

169. ‘Reporting year 2019: school workforce in England’, National Statistics, June 2020; P Tait, ‘After the reboot: computing education in UK schools’, The Royal Society, November 2017.↩

170. P Tait, ‘After the reboot: computing education in UK schools’, The Royal Society, November 2017.↩

171. PEJ Kemp and M Berry, ‘The Roehampton annual computing education report pre-release snapshot from 2018’, May 2019.↩

172. P Tait, ‘After the reboot: computing education in UK schools’, The Royal Society, November 2017.↩

173. S Furber, ‘Shut down or restart? The way forward for computing in UK schools’, January 2012.↩

174. B Lockwood and R Cornell, ‘School ICT infrastructure requirements for teaching computing: a Computing at School (CAS) whitepaper’, Computing at School, 2013.↩

Disabled people

Driving and transport

Education and learning

Employing people

Environment and countryside

Housing and local services

Money and tax

Passports, travel and living abroad

Visas and immigration

Working, jobs and pensions

How government works

Get involved

Help Privacy Cookies Accessibility statement Contact Terms and conditions Rhestr o Wasanaethau Cymraeg

Government Digital Service