COMPUTING AT SCHOOL
EDUCATE · ENGAGE · ENCOURAGE
In collaboration with BCS, The Chartered Institute for IT

Naace

Computing in the national curriculum

A guide for primary teachers

Computing in the national curriculum

A guide for primary teachers

Foreword

Computers are now part of everyday life. For most of us, technology is essential to our lives, at home and at work. 'Computational thinking' is a skill children must be taught if they are to be ready for the workplace and able to participate effectively in this digital world.

The new national curriculum for computing has been developed to equip young people in England with the foundational skills, knowledge and understanding of computing they will need for the rest of their lives. Through the new programme of study for computing, they will learn how computers and computer systems work, they will design and build programs, develop their ideas using technology and create a range of content. But what does this mean for primary schools? How should school leaders be planning for the new curriculum and how can teachers develop the additional skills they will need?

The programme of study is expressed in precise but perhaps unfamiliar language. This guide has been written especially for primary teachers, to demystify the programme of study for primary schools. It will enable teachers to get to grips with the new requirements quickly and to build on current practice. It includes help for schools with planning and gives guidance on how best to develop teachers' skills.

The new national curriculum for computing provides schools with an exciting opportunity to reinvigorate teaching and learning in this important area of the curriculum. We hope this guide will help you on your way.

To find out more about Computing At School, please visit us at
www.computingatschool.org.uk/primary

You will also find an eBook version of this guide there, which can be freely shared with colleagues.

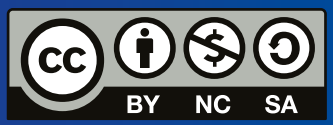
Simon Peyton-Jones
Chairman, Computing At School

Published 2013.

Author: Miles Berry.
Consultants: Amanda Jackson, Penny Patterson and Dave Smith of Havering School Improvement Services.
Text design, Typesetting and Cover Design: Burville-Riley Partnership.
Photography: Ron Coello.

Computing at School are grateful to the following contributors: Phil Bagge, Andrea Carr, Emma Davis, Graham Hastings, Lance G. Howarth, Simon Humphreys, Chris Mairs, Joe McCrossan, Simon Peyton-Jones. Thanks to the children and teachers of Ringwood Infants School and Ringwood Junior School, Ringwood, Abbotswood Junior School, Totton and Gordonbrock Primary School, Lewisham.

We would like to acknowledge and thank ARM Holdings and Raspberry Pi Foundation for their kind financial support without which the production of this guide would not have been possible.

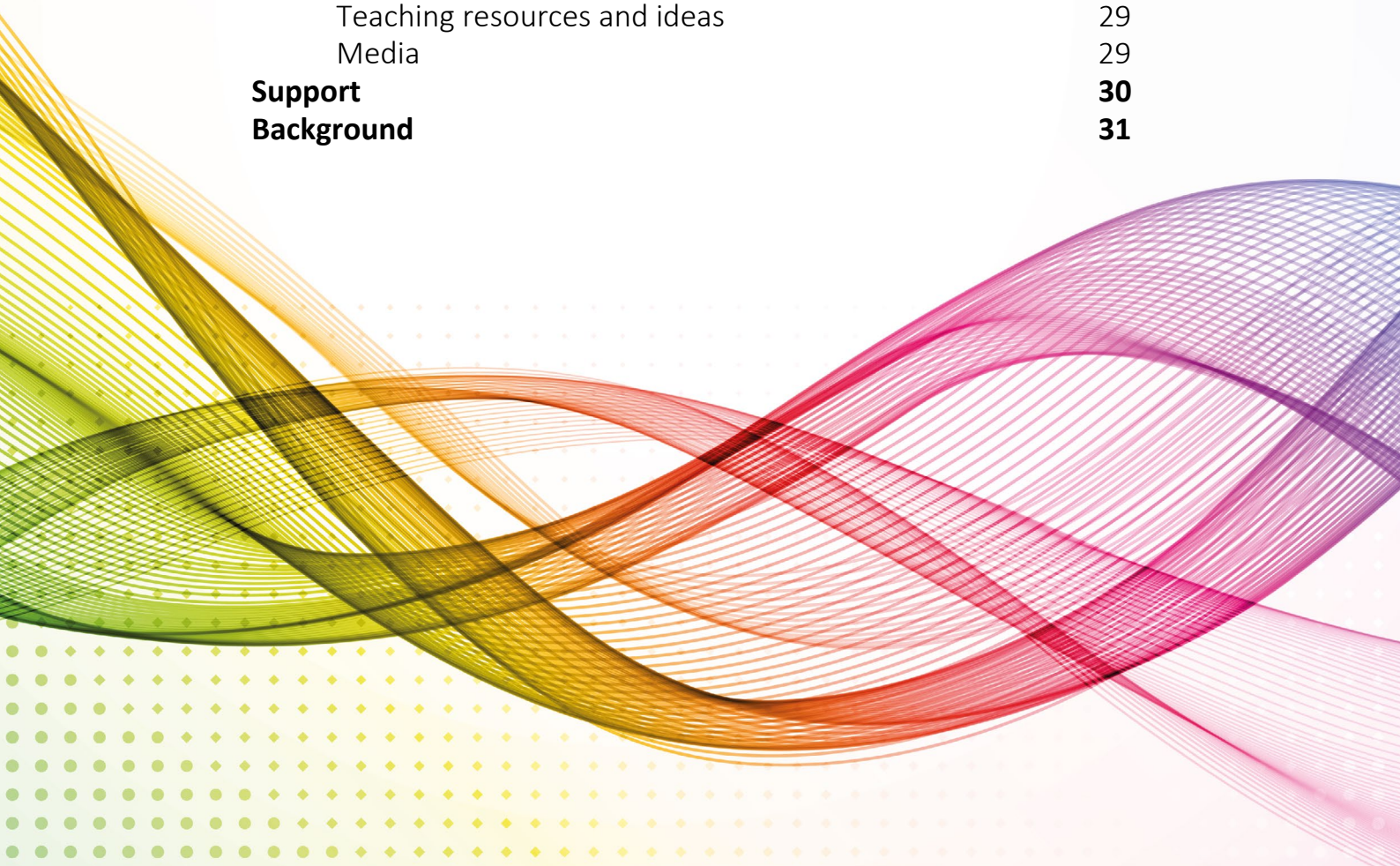


British Library Cataloguing in Publication Data.
A CIP record for this book is available from the British Library.

ISBN: 978-1-78339-143-1

Printed by Newnorth Print, Ltd. Bedford.

Contents



Introduction

The 2014 national curriculum introduces a new subject, computing, which replaces ICT. This represents continuity and change, challenge and opportunity. It gives schools the chance to review and enhance current approaches in order to provide an even more exciting and rigorous curriculum that addresses the challenges and opportunities offered by the technologically rich world in which we live.

Computing is concerned with how computers and computer systems work, and how they are designed and programmed. Pupils studying computing will gain an understanding of computational systems of all kinds, whether or not they include computers. Computational thinking provides insights into many areas of the curriculum, and influences work at the cutting edge of a wide range of disciplines.

Why is computational thinking so important? It allows us to solve problems, design systems, and understand the power and limits of human and machine intelligence. It is a skill that empowers, and one that all pupils should be aware of and develop competence in. Pupils who can think computationally are better able to conceptualise, understand and use computer-based technology, and so are better prepared for today's world and the future.

Computing is a practical subject, in which invention and resourcefulness are encouraged. The ideas of computing are applied to understanding real-world systems and creating purposeful products. This combination of principles, practice and invention makes computing an extraordinarily useful and intensely creative subject, suffused with excitement, both visceral ('it works!') and intellectual ('that is so beautiful').[1]

The focus of the new programme of study undeniably moves towards programming and other aspects of computer science. Programming has been part of the primary national curriculum right from the start, as '**control**' or 'sequencing instructions', although this has too often been overlooked or treated superficially.

There is more to computer science than programming, though. It incorporates techniques and methods for solving problems and advancing knowledge, and includes a distinct way of thinking and working that sets it apart from other disciplines. Every core principle can be taught or illustrated without relying on the use of a specific technology.

The role of programming in computer science is similar to that of practical work in the other sciences – it provides motivation, and a context within which ideas are brought to life.

Information technology deals with applying computer systems to solve real-world problems. Things that have long been part of ICT in schools, such as finding things out, exchanging and sharing **information**, and reviewing, modifying and evaluating work, remain as important now, for a broad and balanced technological education, as they ever were. The new programme of study provides ample scope for pupils to develop understanding, knowledge and skills in these areas, as you'll see from some of the examples in this guide.

Primary teachers currently equip pupils with high-level skills in using ICT, preparing them to apply these across the curriculum in secondary education. It's unclear whether pupils leave primary school with much knowledge of how computers, **software**, the **internet**, the web and search engines work, or a critical understanding of the impact of these technologies on their lives and on society.

As teachers, we are competent and confident users of technology in our own personal and professional lives, and yet relatively few of us are sure how the software running on our computers works, what the difference is between the web and the internet, or how search results are ordered, and we're even less sure of how to teach these things to our pupils. However, with help from the web, new publications and resources, and colleagues (and pupils!) willing to support us, it is time to give it a go.

Note: throughout the guide we have highlighted computing terms in orange. The definitions of these terms are in the glossary on page 27.

[1] **Adapted from *A Curriculum Framework for Computer Science and Information Technology*:**
www.computingatschool.org.uk/data/uploads/Curriculum%20Framework%20for%20CS%20and%20IT.pdf

Getting started

As with other subjects in the new national curriculum, the programme of study document for computing[2] begins with a brief introduction. It presents the subject as one lens through which pupils can understand the world. There is a focus on computational thinking and creativity, as well as opportunities for creative work in programming and digital media.

The introduction also makes clear the three aspects of the computing curriculum: computer science (CS), information technology (IT) and digital literacy (DL).

The core of computing is **computer science**, *in which pupils are taught the principles of information and computation, how digital systems work and how to put this knowledge to use through programming. Building on this knowledge and understanding, pupils are equipped to use* **information technology** *to create* ***programs***, *systems and a range of content. Computing also ensures that pupils become* **digitally literate** *– able to use, and express themselves and develop their ideas through, information and communication technology – at a level suitable for the future workplace and as active participants in a digital world.*

One way of thinking about these aspects is as the foundations, applications and implications of computing. The aims for the subject as a whole reflect this distinction.

[All pupils] can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation. ***(CS)***

[All pupils] can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems. ***(CS)***

[All pupils] can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems. ***(IT)***

[All pupils] are responsible, competent, confident and creative users of information and communication technology. ***(DL)***

It's worth noting that computer science aims to cover two distinct, but related, aspects. There's a focus on computer science itself (the ideas and principles that underpin how digital technology works) but this sits alongside the practical experience of programming, almost certainly the best way for primary pupils to learn about computer science.

Your school has a statutory duty to offer a broad and balanced curriculum that prepares pupils to 'use computational thinking and creativity to understand and change the world'.[3] Therefore, as your school develops its scheme of work for computing, it would be unwise to ignore any of these aspects, or to give too much emphasis to one to the detriment of the others.

That said, you have the freedom to decide how much time you spend on any aspect of the programme of study, and there's no implication that the number of bullet points or words should be proportional to the time spent on any aspect, as long as pupils have been taught all the content by the end of the key stage.

[2 and 3] **See www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study**

We will look in more detail at the programme of study, but a quick scan of the subject content shows expectations for the three aspects of computing at each key stage. The content has been adapted below to show how it can be broken down into three sub-sections.

	KS1	KS2
CS	Understand what algorithms are; how they are implemented as programs on digital devices; and that programs execute by following precise and unambiguous instructions Create and debug simple programs Use logical reasoning to predict the behaviour of simple programs	Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts Use sequence, selection, and repetition in programs; work with variables and various forms of input and output Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs Understand computer networks including the internet; how they can provide multiple services, such as the World Wide Web Appreciate how [search] results are selected and ranked
IT	Use technology purposefully to create, organise, store, manipulate and retrieve digital content	Use search technologies effectively Select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information
DL	Recognise common uses of information technology beyond school Use technology safely and respectfully, keeping personal information private; identify where to go for help and support when they have concerns about content or contact on the internet or other online technologies	Understand the opportunities [networks] offer for communication and collaboration Be discerning in evaluating digital content Use technology safely, respectfully and responsibly; recognise acceptable/unacceptable behaviour; identify a range of ways to report concerns about content and contact

It should be noted that the statutory requirements are not labelled under these three headings in the programme of study, and the distinction between information technology and digital literacy is open to some interpretation. The important thing is to cover the content in a balanced, stimulating and creative way rather than being overly concerned about the specifics of terminology.

There are big changes in assessment, too, as with other subjects of the national curriculum. The old system of levels will be abolished and is not being replaced. How your school chooses to assess, record and report pupils' mastery of the curriculum content is your decision, but we explore some possible approaches in the Assessment section.

Subject knowledge

The statements in the programme of study are brief. Let's take a slightly more detailed look at the concepts each statement refers to.

Key stage 1

Understand what algorithms are; how they are implemented as programs on digital devices; and that programs execute by following precise and unambiguous instructions

An **algorithm** is a precisely defined procedure – a **sequence** of instructions, or a set of rules, for performing a specific task (e.g. instructions for changing a wheel or making a sandwich). While all correct algorithms should produce the right answer, some algorithms are more efficient than others. Computer scientists are interested in finding better algorithms, partly out of intellectual curiosity, and partly because improvements in algorithms can result in massive savings in terms of both cost and time.

Computer programs, like algorithms, are comprised of sets of rules or instructions, but they differ in that they need to be written in a precise language a computer can 'understand'. A computer's central processor understands a very limited set of simple instructions written in *machine code*. Very few programmers work at this level, so computer scientists have developed *programming languages,* which sit somewhere between the ideas in the algorithm and the computer's machine code.

A programmer can turn an algorithm into code using a programming language that has enough in common with the English language to make it easy to read, remember and write. The programming language takes care of the minute details, like how to do multiplication or where **data** should be stored in the computer's memory, which means the programmer can focus on the big picture.

There are many different programming languages. They each have their own vocabulary, grammar and features that make them appropriate for particular tasks. The current favourites in primary schools are Scratch, Logo and Kodu.

Programs are made up of statements in a limited, but precisely understood, vocabulary. Each statement in the program has one particular meaning. The computer follows the instructions given: nothing more and, almost always, nothing less.

A 'computer' is not just a traditional desktop or laptop PC; it is any device that accepts **input**, processes it according to a stored program, and produces an **output**. The input, stored program and output are all encoded as numbers, making these devices 'digital'. Digital devices include the controller in your car or microwave oven, your mobile phone, tablet, laptop and desktop, as well as high-end supercomputers and 'virtual' servers in the 'cloud'.

Create and debug simple programs

The best way for pupils to learn what an algorithm is, and how it can be implemented as a program, is to write some programs themselves. Programming involves taking an idea for doing something and turning it into instructions the computer can understand. In the infant classroom this could be writing a set of commands for a Bee-Bot, Pro-Bot or Roamer, or snapping on-screen program building blocks together in Scratch.

When you write a program you need to have a clear idea of what it will do and how it should do it. This is where algorithms come in, and thinking algorithmically is an integral part of the craft of programming.

Most programs don't work as they should first time round; professional programmers have this experience all the time! One of the most rewarding aspects of programming is finding and fixing these mistakes. Mistakes in programs are called 'bugs', and finding and fixing them is '**debugging**'.

The process of debugging often involves identifying that there is a fault, working out which bit of the program (or underlying algorithm) has caused the problem, and then thinking logically about how to fix it. In the classroom, this can provide a great opportunity for collaborative work.

As a teacher, you should identify clear steps that pupils can follow so that they can fix their code. These might involve identifying what the fault is, finding out which part of the code is creating the problem, and then working towards a fix.

Pupils should be encouraged to work together to identify bugs, as programmers are often blind to their own mistakes. Although it might be appropriate to help pupils compare code or identify which section to look at, it is rarely helpful for you to fix a bug for pupils until they have worked through the stages of debugging themselves. Debugging code develops valuable learning skills that are transferable right across the curriculum, such as independence, resilience and persistence.

Use logical reasoning to predict the behaviour of simple programs

Computers are deterministic machines. We can predict exactly how they'll behave through repeated experience or by developing an internal model of how a piece of software works. Stepping through the program can give a clear sense of what it does, and how it does it, giving a feel for the algorithm that's been implemented.

In the classroom, getting one pupil to role-play a floor turtle or screen sprite while another steps through the program can give a far more immediate sense of what's going on. When working with a computer, encourage pupils to make a prediction about what the program will do before they press return or click the button, and to explain their prediction logically; this is part of computer *science*.

Logical reasoning also implies that pupils are following a set of rules when making predictions. Pupils who step outside the boundaries of these rules are not using logical reasoning. A pupil who expects a roamer to jump doesn't understand the constraints of its programming language or hardware.

Use technology purposefully to create, organise, store, manipulate and retrieve digital content

Creating digital content has many practical possibilities. These include commonplace tasks such as word-processing, creating pictures using paint packages, working with digital photographs and video (including animations), writing computer programs, and creating online content such as blog posts, forum contributions, wiki entries and social network updates. This creative work is digitised (i.e. converted to numbers) once it's on the computer.

The sheer quantity of digital information makes the skill of **organising** digital content more important than ever. In more practical terms, we might think of how to bring together different digital media, how to order a series of paragraphs, how to organise the files in our documents directory, or how to tag photos and posts online.

Storing digital content is perhaps something we take for granted. Knowing where a file is saved in the directory structure is important. It's vital to be able to distinguish between the hard disk (or solid state storage) inside the computer itself, the school's network server, USB disks or memory cards, and online storage via the internet.

Content is stored digitally. Size is measured in bytes, one byte being the amount of information needed to encode a single character of text. A kilobyte (kB) is 1000 bytes, 1000 kB is a megabyte (MB), 1000 MB is a gigabyte (GB) and 1000 GB is one terabyte (TB). The list continues beyond that. A short word-processed document might be 25 kB, a digital photo 5 MB, a feature-length, high-definition film 4 GB and the data on a computer hard drive 1 TB.

Manipulating digital content is likely to involve using one or more application programs, such as word-processors, presentation software, or image-, audio- or video-editing packages. The pupil makes changes to the digital content, which might include combining content from multiple sources. The skill here is not just using the software tools, but also knowing how best to change the content for the audience and purpose, and to take into account principles of good design.

Retrieving digital content could be seen as the reverse of storing: the skills of opening and saving documents are similar. Retrieving content requires you to know what you called the file, what file type it is, and where you stored it.

Finding files can be time-consuming, especially when the filing system is not well organised. Computer filing systems have **search** features to make this easier, but are reliant on the user

remembering enough about the file to be able to search for it. The problem of finding a particular file is harder on the web, although the links between web pages help, and these are at the centre of Google's algorithm for ranking search results.

Recognise common uses of information technology beyond school

Digital technology is a part of all our lives, with almost no sphere untouched by it. A key stage 1 pupil might be woken by a digital alarm clock, have a bowl of microwaved porridge for breakfast, and then watch digital TV or play an iPad game before travelling to school, their journey guided or tracked via GPS.

While they're at school, their attendance, progress and lunch are tracked through the management information system, they engage in activities on tablets, and research things on the web. Their parents use digital technology at work, perhaps using computerised control and monitoring equipment in manufacturing, productivity suites in an office, or high-end digital tools in creative industries.

The ingredients for the evening meal may have been ordered online, or a parent may have scanned them at the supermarket, whose supply chain is controlled by smart systems. Evening entertainment might be computer gaming with a Wii or Kinect. Parents and older siblings socialise on smartphones or laptops, and the book at bedtime might be read on an e-book reader.

There are many opportunities for pupils to consider the applications of algorithms, programs and systems.

Use technology safely and respectfully, keeping personal information private; identify where to go for help and support when they have concerns about content or contact on the internet or other online technologies

This statement covers the key principles of pupils' e-safety. Pupils should be aware of the main risks associated with the internet, and recognise that they should not share certain types of personal information online.

Young children have little awareness of who can access online information, so it is best to teach them not to communicate any personal information online. Pupils should develop their sensitivity to others online, treating them with respect, and showing respect for their privacy.

Pupils should have an age-appropriate understanding of their responsibilities under the school's acceptable use policy. As pupils may inadvertently access inappropriate content on the web, they need to know how to report a worry, and they should be encouraged to talk to teachers or parents about their concerns.

Adults worry about extreme content, but pupils' worries are often at a lower level, related to material they consider unfair or unkind. In order for pupils to feel supported, it is important that adults empathise with, and address, these worries, and there should normally be no blame attached to a pupil reporting such concerns.

Pupils must have a clear understanding of what to do if they have concerns about inappropriate online behaviour (such as unwelcome contact or cyberbullying). Telling a teacher or parent should normally be the first response, but pupils should also know that they can talk directly and confidentially to Childline about such matters.

You must follow your school's child protection policy, and your child protection lead must be informed about any potential abuse, whether online or offline. This may include informing the Child Exploitation and Online Protection Centre (CEOP).[4] Further information for teachers on e-safety is available on CEOP's Thinkuknow[5] site.

[4] See www.ceop.police.uk

[5] See www.thinkuknow.co.uk/teachers/

Key stage 2

Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts

The focus on algorithms at key stage 1 leads pupils into the design stage of programming at key stage 2. Algorithms are the necessary start of the process of creating working code, and identifying the steps needed to solve any problem is essential.

Splitting problems into smaller parts is part of computational thinking. For example, designing a game in Scratch will involve thinking about algorithms, programming, drawing sprites and backgrounds, making animations, and even composing music or recording sound effects.

We think of computers as boxes with keyboards, mice and displays, but built-in computers (or 'embedded control systems') are an increasingly significant application of information technology. Pupils can gain valuable insights into how computers are used to monitor and control real-world systems by using sensors, switches, motors and lights. Computers also make it possible to explore real-world situations that would be too difficult, too expensive or too dangerous to create in real life.

Use sequence, selection, and repetition in programs; work with variables and various forms of input and output

Sequence in this context is the step-by-step nature of computer programs, mirroring the sequence of steps the algorithm would list.

Selection refers to instructions such as *if ... then ... otherwise* decisions in which the operation (what the program does) depends on whether or not certain conditions are met. For example, a quiz provides different feedback if the player answers the question correctly or incorrectly. It is helpful to refer pupils to selections (choices) they make in everyday life; for example, *if* it rains in the morning, *then* I will wear my anorak to school, *otherwise* I won't.

Repetition is a programming structure such as a *repeat ... until* loop in which the computer runs part of the program a certain number of times or until a particular condition is met.

In the case of the quiz, we might want to ask ten questions, or keep going until the player has scored five correct answers. Again, it is useful to refer pupils to loops or repetition in daily routines. For example, the traffic lights on a pelican crossing will stay green *until* someone presses the button to cross the road; an oven heats up *until* it reaches the right temperature. There are many loops in the wider world, such as the days of the week or the moon travelling around the Earth.

Variables are used to keep track of the things that can change while a program is running. They are a bit like *x* or *y* in algebra, in that the values may not initially be known. Variables are not just used for numbers. They can also hold text, including whole sentences ('strings'), or the logical values 'true' or 'false'. For our quiz we would use variables to keep track of the player's score and the number of questions they attempt. Variables are like boxes, in that the computer can use them to store information that can be changed by the user, the program or by another variable.

We may think of **input** as keyboard and mouse (or touch screen), and **output** as the computer display, but pupils' experiences should be widened beyond this. Working with sound is straightforward, as laptops have built-in microphones and speakers. The latest version of Scratch provides support for using webcams. Digital cameras allow interesting work using image files.

The reference in the programme of study to 'controlling physical systems' implies the use of sensors, motors and perhaps robotics. Midi instruments like an electronic keyboard, and devices such as MaKey MaKey[6] and Microsoft Kinect provide yet further experience of working with various forms of input.

Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs

Key stage 2 pupils should be able to explain the thinking behind their algorithms, talking through the steps and explaining why they've solved a problem the way they have. They also need to be able to look at a simple programming project and explain what's going on. This is made easier with languages like Scratch, Kodu and Logo, which feature an on-screen sprite or turtle. The immediate feedback helps pupils to understand and debug their programs. Pupils might also be expected to look at someone else's algorithm and explain how it does what it does.

Thinking through programs and algorithms helps develop pupils' abilities to think logically and algorithmically, which leads to planned debugging of code rather than just a trial-and-error approach.

Understand computer networks, including the internet; how they can provide multiple services, such as the World Wide Web, and the opportunities they offer for communication and collaboration

This is a challenge because most of us have not thought about how these ever-present technologies do what they do.

Computer networks, including the internet, are made up of computers connected together. The computers include fast, dedicated machines that pass on data that's not intended for them (called 'routers', 'gateways', 'hubs' or 'switches', depending on particular roles), and 'servers' (always-on machines looking after emails, web pages and files that other computers might ask for from time to time). The connections between the computers in a network may consist of radio or satellite signals, copper wires or fibre-optic cables.

Information stored on computers and information travelling over networks must be digitised (i.e. represented as numerical data). The computer network in your school and the internet use the same method or 'protocol' to send and receive this data. The data is broken up into small 'packets', each with identifying information, which includes the IP (internet protocol) address of the sender and recipient.

These packets of information make their way across the internet from source to recipient. At the far end, the packets get stitched back together in the right order and the email is delivered, the website is accessed, or the Skype call gets connected. Many of these packets, travelling at near light-speed, are generated by web servers returning web pages to the browser requesting them.

By connecting people around the world and passing on packets of data from sender to recipient, the internet has created many opportunities. These range from communication (such as email, video conferencing, blogs, forums, social networks) and collaboration, such as wikis (including Wikipedia), to real-time collaborative editing, Creative Commons media (permission to share and use creative work with conditions stated by the creator) and open-source software, which is available for us to use and change.

[6] See www.makeymakey.com

Use search technologies effectively, appreciate how results are selected and ranked, and be discerning in evaluating digital content

Using search technologies involves aspects of computer science, information technology and digital literacy. Effective use of search engines gets the results you want. It relies on specifying the right keyword, skimming and scanning the results to see which seems most relevant, and distinguishing between the main results and adverts presented as sponsored results. It may also involve using other features[7] of the search engine, including searching for phrases rather than keywords, or limiting searches to a particular time frame, language, reading level or website.

In order to return results, search engines use 'web crawler' programs. These programs visit the pages of the web, follow the links they find and can make a copy of each page visited. The pages are indexed, keeping track of keywords on each page. When you enter a search query, the search engine returns pages from its index on which your keyword(s) or phrase appears.

Search engines take many factors into account. At the heart of Google's algorithms[8] is 'PageRank', which determines the quality and rank of a page based on the quality of the pages that link to it. Their quality is, in turn, determined by the quality of the pages that link to them, and so on.

Just because a page has a high rank in Google or another search engine for a particular query, it doesn't mean that the content is true, age-appropriate or relevant to a particular project. Pupils need to develop skills in evaluating digital content, including how trustworthy the information is (perhaps by verifying it with another independent source), whether it's something that the audience for a project would be able to grasp, and why the content was posted in the first place (e.g. to give a balanced overview, or simply to advance one side of an argument).

[7]See, for example, www.google.com/advanced_search

[8]There's an overview of some of Google's algorithms at www.google.co.uk/intl/en/insidesearch/howsearchworks/algorithms.html

Select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information

This is something of a catch-all requirement, bringing together various aspects of the computing curriculum. Pupils might typically be expected to demonstrate progression by:

- using software under the control of the teacher
- then, using software with increasing independence
- then, combining software (e.g. importing an edited image or video into a presentation or web page)
- then, selecting software themselves (perhaps from the full range of applications installed on computers, smartphones and tablets at home or at school, or available to them via the web).

Internet services might include, for example, learning platforms, school, class or individual blogs, and cloud-based tools such as Google Drive, Office 365 or image-editing sites.

The reference to 'a range of digital devices' encompasses using both fixed and mobile technologies. It also includes running software (such as that described in the previous paragraph) on web servers via the internet.

There is also recognition that design and creativity in computing encompass many forms, from the content familiar to many from the old ICT programme of study, the programming as required by earlier statements in the new programme of study, to more complex, system-level ideas, combining software and hardware to achieve a well-defined goal with a particular audience in mind.

There is an important distinction between data and information at GCSE and A level, where information is defined as structured data that has been processed and has meaning attached to it. At key stage 2 it might be more helpful to think of data as numbers and information as richer media such as text, images, audio, and video or 3D representations. However, it is worth remembering that both data and information are digitised by computers (i.e. stored in the form of numbers).

Collecting, analysing, evaluating and presenting data is an important application of computers. Pupils should gain experience of working with data they have generated or collected for themselves, as well as big, public datasets.[9]

Pupils have an opportunity to develop a more critical media literacy as they work with tools that, until relatively recently, were the domain of professionals. Tools for recording audio and video, and for creating animation, web pages, digital photos, digital music and 3D models, are all available to primary schools for low (often zero) cost. Providing a potentially global audience for the pupils' work is tremendously motivating.

Use technology safely, respectfully and responsibly; recognise acceptable/ unacceptable behaviour; identify a range of ways to report concerns about content and contact

Safe and responsible use of technology at key stage 2 builds on skills learned in key stage 1. As well as requiring pupils to keep themselves safe and to treat others with respect, the programme of study at key stage 2 introduces an emphasis on *responsible* use of technology.

Pupils need to consider how their online actions impact other people. They need to be aware of their legal and ethical responsibilities, such as showing respect for intellectual property rights (e.g. musical, literary and artistic works), keeping passwords and personal data secure, and observing the terms and conditions for web services they use (such as the 13+ age restriction on most US websites, including Facebook, resulting from COPPA[10] legislation).

Pupils should also develop some awareness of their digital footprint: the data automatically generated when they use the internet and other communication services, and how this is, or could be, used.

Pupils should be aware of, and abide by, the school's acceptable use policy, as well as the requirements of any other services they use. Encourage pupils to think twice, and to check terms and conditions, before signing up for internet-based services.

As in key stage 1, pupils should report any concerns to a parent or teacher. They should also be aware that they can talk directly to the police, report their concern to CEOP, or talk in confidence to counsellors at Childline. Your designated child protection lead might, depending on the nature of the concern, raise the matter with local social services, the police or CEOP.

[9]See http://data.gov.uk/ and www.theguardian.com/news/datablog/interactive/2013/jan/14/all-our-datasets-index

[10]The Children's Online Privacy Protection Act, which prohibits companies in the United States from storing any information on under 13s: see www.coppa.org/coppa.htm

Planning

How can we turn the requirements of the programme of study into engaging lessons?

Here are four things to keep in mind.

- The programme of study is a *minimum* entitlement – there's nothing that imposes any limits on what schools, teachers or pupils can cover in computing.
- The programme of study is not a scheme of work – it's up to you, as a school, to determine how you cover this content, in what order, in what contexts and with what resources.
- Schemes of work are not lesson plans – that level of planning comes later, with the ideas for each unit of work getting translated into the detail of specific objectives, resources, activities and assessment.
- There is a far greater focus now on learning about computers and computation, not simply learning how to use technology.

The opportunity to do something really creative is there for the taking. A number of strategic decisions need to be made before work can properly begin on developing a scheme of work for computing, and it would be wise to consult with stakeholders and potential partners before committing to any one path. You'll need to consider the following areas.

Discrete or embedded? There were strong arguments for adopting an 'embedded' approach to the old ICT programme of study, in which ICT capability was covered in meaningful contexts derived from other subject areas.

This looks to be a lot harder for computing because of the discrete subject knowledge expectations, but it's certainly not impossible. In fact, there are wide applications of computational thinking (such as looking at algorithms and decomposing problems into smaller steps) across the curriculum, and there's plenty of scope for using other subject areas to provide interesting objectives for pupils' programming projects.

National curriculum or national curriculum 'plus'? Remember that the national curriculum is the minimum. Will you choose to include additional content? If so, what other things might be added to the list? There are arguments that the key stage 2 curriculum should also include an *explicit* requirement for creative work with html. Note that these things are not prohibited, and you might like to include these, or other, elements when developing your own scheme of work.

Themes? As you read through the programme of study, what overarching themes suggest themselves to you? Do these provide a structure that ensures both progression and continuity as pupils move through primary school? Might these be one way of fitting different parts of the computing curriculum together?

Grid? How detailed should the scheme of work be? Many schools adopt a half-termly grid, but a more flexible structure might suit your school better. Similarly, consider whether the scheme of work needs to specify the order in which each year's units are studied. Is the order important for progression? Should individual class teachers be able to decide?

Format? In practical terms, what should the final document look like? A single table, tables for each year/half term, or simply text laid out in paragraphs? Will you need to print a copy or can it be entirely online, perhaps as a collaborative document (e.g. in Google Drive, a wiki or on GitHub[11]) for you and your colleagues to revisit and revise in the light of the experience gained from teaching it.

[11] See https://github.com/

Also think about how much detail needs to be specified – as a rule of thumb, include enough for a teacher lacking in confidence to feel that they can do a good job, but not so much that the most confident feel limited by what's there. Depending on the decisions above, it would be reasonable to expect a scheme of work to include:

- topic title
- curriculum coverage
- learning objectives
- outline of activities
- resources
- cross-curricular links
- assessment opportunities.

There are several ways to go about implementing a scheme of work for computing.

- *Top down*, starting from the programme of study itself.
- *Bottom up*, starting with ideas for projects and units of work, which include cross-curricular and embedded approaches.
- *Off the shelf*, using a commercial, free or crowd-sourced scheme, perhaps with some modifications.
- A more *pupil-centred, enquiry-led approach*, although a scheme of work in this context might merely suggest possible projects, resources and a consistent approach to monitoring achievement and curriculum coverage.

Let's consider these approaches in turn.

Starting with the programme of study

The programme of study gives a clear list of the content that should be covered in each key stage, to which you might like to add further elements of your own. One advantage of using the programme of study as your starting point is that it's relatively easy to translate the content into specific objectives, because it's clear what needs to be covered, and when.

In planning a scheme of work, it's sensible to look for themes that can provide a structure, making it easier to ensure progression and continuity over the time a pupil is at primary school.

Perhaps the most obvious set of themes is computer science, information technology and digital literacy. You could further divide the computer science aspect into *Programming* and *Other elements of computer science*. The 'foundations, applications, implications' characterisation of these elements would provide a similar overall structure.

Another approach identifies six aspects: *Coding, Computer science, Networks and the internet, Communication and collaboration, Creativity* and *Productivity*. This leads to a half-termly grid, with each aspect being the focus for half a term.

A fourth option might be to look beyond the computer science/information technology/digital literacy taxonomy to broader themes across the subject. Dividing the curriculum into *Computational thinking, Design* and *Criticality and responsibility* would be one approach.

Whichever themes you select, revisiting these areas in each year can ensure both continuity and progression for pupils, and make it easier to plan individual units of work. There should be a clear sense of what pupils have already experienced, and what subsequent steps in learning are likely to involve. A whole-school programming strand might look something like this.

Year 1	Solving problems with Bee-Bots
Year 2	Turtle graphics on the floor and screen
Year 3	Scripted animations
Year 4	A maths quiz
Year 5	Computer games
Year 6	Developing applications for the mobile phone

The above is intended for illustration only. It's important to remember that the focus is on developing an understanding of programming, rather than developing skills in using just one programming language. A similar sequence of half-termly units could be developed for other themes.

Starting with projects

An alternative approach is to start with ideas for individual projects, perhaps with each being half a term in length.

If this is the approach you've used until now for planning ICT, you might find that you can use many of your existing ICT projects to cover some of the computing curriculum. This is particularly true for the information technology and digital literacy elements, although you may have to make some changes to allow space for the new computer science content, including the additional expectations for programming.

Some of your existing units will perhaps need modifying to focus on knowledge and understanding rather than skills. For example, if you have a current unit on email, that could be modified to develop pupils' understanding of how networks, including the internet, work, how they provide services such as email, and how this can be used for communication and collaboration. You could also cover key issues in e-safety such as spam, malware in attachments, and spoofed links.

A project-based approach allows ample scope for exploiting the connections between the different aspects of computing, perhaps using the 'foundations, applications, implications' model as a starting point for planning some of these units of work.

Projects could be linked to other areas of the curriculum, perhaps using themes from your school's 'creative curriculum' to suggest related computing topics. Similarly, this approach would work if you've decided to adopt an embedded or integrated approach to computing, with computing content covered through topics drawn from other curriculum areas.

For example, there are links between algorithms and maths. Creating a Scratch script for a maths game that tests a player on adding fractions would develop an understanding of the algorithm for fractions, as well as the sequencing, selection, repetition and variables requirements of the computing programme of study.

[12] See http://community.computingatschool.org.uk/door
[13] See http://tryruby.org/levels/1/challenges/0
[14] See www.codecademy.com
[15] See www.khanacademy.org/science/computer-science

Using a pre-written scheme of work

It is likely that various organisations and individuals will develop schemes of work for the new computing curriculum. It's perhaps preferable to think in terms of adapting, rather than adopting, schemes of work developed by others, whether commercial or otherwise. A sensible approach would be to use an existing plan as a starting point, and then edit it so that it draws on the expertise and enthusiasm of your colleagues, fits well with other areas of your curriculum, makes use of the resources you have and, vitally, appeals to your pupils.

The internet, of course, makes it easy to collaborate on documents, so there's no need to do all this planning on your own. Joining with like-minded colleagues in a local network, via a subject association, or in informal groups via Twitter or other social networks, will allow you to draw on others' insights and experience, and your contribution may impact on pupils' learning beyond your own school. The Computing at School (CAS) Community[12] is a good resource – either online or through its network of local hubs.

Using a pupil-centred approach

One way of going about this might be to develop a set of modular projects for pupils to choose from, structured so that there is a clear progression from easier to harder projects.

Another pupil-centred method would be to use an enquiry-based approach at the beginning of each half-termly unit: briefing pupils on the overall topic, and then establishing what they already know and what they'd like to find out. The unit can then be planned in detail around areas of particular interest to pupils.

There are ample resources available to support a more independent approach to learning computing. Scratch and Kodu have vibrant user communities; online interactive tutorials provide an introduction to programming languages such as Ruby[13] and Javascript,[14] and there are many tutorials and walkthroughs on Khan Academy[15] and YouTube.

Resourcing

Alongside any curriculum development work, some thought needs to be given to providing the resources necessary for teaching. Despite the opportunity to use resources like the excellent materials provided by New Zealand-based CS Unplugged,[16] you will probably need a set of computers for teaching computing. General-purpose laptops and desktops are ideal, and it really doesn't matter if you're using Windows PCs or Macs, or even Linux, for the primary computing curriculum. The Raspberry Pi offers a great platform for programming and developing pupils' understanding of networks and the web.

For many activities, pupils may need access to the internet, particularly the web. You'll need to make sure the usual safeguards are in place, but Ofsted's recommendation[17] is not to be too restrictive; they advocate a managed, rather than a 'locked down', approach. They recommend that pupils need to learn how to use technology safely, respectfully and responsibly, not to have their responsibility for this taken on by others.

You'll need some tools with which pupils can program their computers. MIT's Scratch, for example, provides all the tools needed to cover the programming requirements of the new curriculum.

Alternatives are available: Kodu is a rich, game-like environment providing a graphical 'way in' to programming; Logo has a very long history as an introductory programming language, although as it's text-based there's plenty more scope for bugs in code through typing or spelling errors. Some leading primary practitioners are introducing pupils to text-based programming using Python.

While the programming expectations for key stage 1 can be met using screen-based programming tools such as Scratch, there's much to be said for working with programmable toys at this age, such as Bee-Bots, Roamers, Pro-Bots and Big Traks, although there's certainly no requirement to do so.

At key stage 2, if you want to go down the 'controlling physical systems' route, you'll need some cheap components (sensors, lights and motors) and some way of connecting these to a computer. The FlowGo interface can be used with Windows PCs. LEGO®'s WeDo interfaces nicely with Scratch 1.4, and there are interesting, perhaps more demanding, possibilities using platforms such as Arduino or Raspberry Pi.

At key stage 2, pupils are expected to use other digital devices, which could be as simple as digital cameras or audio recorders, but could also include more complex devices such as smartphones or tablets. There is also an expectation that pupils will have access to internet-based services, such as the school's learning platform, a blog, or cloud-based software such as Google Drive or Office 365.

Many schools are considering providing pupils with access to tablets. They can enhance learning across the curriculum, particularly if coupled with corresponding pedagogic developments. Although tablets were not intended as a programming platform, there are a growing number of apps[18] that provide an introduction to programming. It's also possible to access HTML5-based online programming tools such as Snap![19]

[16]See www.csunplugged.org

[17]See www.ofsted.gov.uk/sites/default/files/documents/surveys-and-good-practice/t/The%20safe%20use%20of%20new%20technologies.pdf

[18]See http://antsict.wordpress.com/2013/02/23/coding-computer-science-and-ipads-my-current-view/ for a good overview.

[19]See http://snap.berkeley.edu/snapsource/snap.html, a close variant of Scratch.

Teaching

Seymour Papert (1928–) is seen by many as the pioneer of computing in schools. He is probably best known as the co-developer of the Logo programming language in the late 1960s.

Logo introduced the idea of turtle graphics, in which a computer-controlled robot 'turtle', equipped with a pen, moves, turns and draws to make shapes on paper. A child who is programming Logo can define their own 'words' (procedures) so, for example, the turtle could be programmed to make a square by giving the command, 'Move forward and turn 90°' four times.

Papert saw Logo as more than a programming language, though; he believed it was a powerful tool for pupils to develop their thinking skills.

I began to see how children who had learned to program computers could use very concrete computer models to think about thinking and to learn about learning and in doing so, enhance their powers as psychologists and as epistemologists.[20]

Insights such as this lie at the heart of the changes in the curriculum from ICT to computing. Many teachers may recall Logo from their own school days, and Logo was a key influence on Scratch, which was developed by one of Papert's PhD students.

Inspired by his work with Logo was Papert's theory of learning: constructionism. Put simply, this is the theory that people learn best through making things for other people.

Learning as 'building knowledge structures' . . . happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity.[21]

Pupils learn more when they write about a topic than when they read about it, especially if they know that you, and perhaps others, will be reading what they write. It seems likely that this is true of every aspect of computing.

- Pupils will learn computer science far more effectively by writing programs to show to others.
- Pupils will learn to use information technology more effectively if they're doing something creative, such as making a presentation, website or video, especially if this is to be shown to others.
- Pupils will develop a richer digital literacy if they document what they know and learn for others through blog posts, audio recordings or screencasts.

When teaching the computing curriculum, look wherever you can for practical, creative projects that pupils can work on, perhaps individually, perhaps with a partner, or as part of a small group: this, after all, is how programming and information technology happens in the 'real world' and on most university courses. The projects you set are more likely to be motivating if they're linked to your pupils' own interests and enthusiasms. These might be to do with other curriculum areas, the life of the school, or their interests beyond school.

Also, look for an audience for pupils' work, whether they're presenting to one another, writing for a public blog, creating software or digital content for younger pupils, or planning to upload their work for others to see, via Scratch or a school YouTube account.

[20] **Papert, S., *Mindstorms: Children, Computers, and Powerful Ideas*, (Basic Books, 1993), p.21.**

[21] **See www.papert.org/articles/SituatingConstructionism.html**

Games can be very motivating, and pupils often enjoy evaluating each other's work. Remember, though, that such projects are *not* an end in themselves: the focus should remain on developing knowledge and understanding of computing *through* such activities, however engaging they may be. Your role as a teacher extends beyond setting the challenge and providing support in projects, to helping pupils understand the ideas that lie at the heart of the creative work in which they're engaged, and to helping pupils make the connection between these concepts.

Here are a few examples of projects.

- Making and editing a cookery video in which the algorithm of a recipe is clearly illustrated.
- Creating a video game using characters and settings from a shared reading book.
- Developing educational software for younger pupils to practise mental arithmetic.
- Creating a scripted or stop-motion animation telling the story of an email's journey from sender to recipient.
- Adding content to the Simple English Wikipedia to explain computing concepts (or concepts from other topics pupils are studying) to a global audience.
- Developing a micro-site for the school on how to use the web safely, respectfully and responsibly.

Many other ideas for creative projects will suggest themselves, either starting from, or ending with, the programme of study content.

American educationalist David Jonassen[22] coined the term 'meaningful learning' to describe projects such as these. He identified five essential aspects for learning to be described as 'meaningful', and these might help in considering what makes for effective learning in computing.

Active: Pupils should be actively engaged in their learning – typically this will be *doing* something on a computer, but it could also be taking part in a discussion or an activity away from the computer, such as role-play to illustrate how packets of data travel across the internet.

Constructive: This can be understood both in the sense of constructing meaning, developing pupils' mental model of computation and technologies, and in the sense of making something, whether this is a computer program, a presentation or a blog post.

Intentional: Ideally, pupils should have some degree of *choice* over how they tackle a task or project, or perhaps even over the task or project itself. It is unlikely they will learn much from copying a worked solution off an IWB screen, and many projects can be constructed or adapted to allow plenty of scope for individual creativity.

Authentic: Wherever possible, try to link activities with pupils' own experiences, both within and beyond school: cross-curricular projects work very well, as do those linked to the life of the school itself, or to pupils' experiences of technology.

Cooperative: Computing, in both industrial and academic contexts, is a collaborative endeavour. Where possible, construct activities so that pupils can work together, supporting one another in their learning.

This is not to say that creative, collaborative projects are the only, or in some circumstances even the best, approaches to teaching computing. There are many topics where pupils will learn a lot through classroom discussion, teacher demonstration or watching high-quality media.

[22] **Howland, J. L., Jonassen, D. H. and Marra, R. M., *Meaningful Learning with Technology* (Pearson, 2011).**

Technologically enhanced learning

There are many high-quality, often interactive, resources available via the web to support pupils' learning in computing.

Typing error messages into a search engine will often give a pointer towards a solution, and provide some opportunity for 'just-in-time' learning in the process.

YouTube hosts countless 'walkthrough' tutorials for a wide variety of software packages, including programming toolkits such as Scratch. Your pupils might add their own.

Wikipedia[23] provides comprehensive coverage of computing topics and links for further study, as well as promoting a more thoughtful evaluation of online information and a potential audience for pupils' own contributions.

There are plenty of opportunities for pupils to seek help, get feedback, provide support to others and share their work with an audience beyond the classroom through your school learning platform, web space provided by your local authority or regional broadband consortium, and online communities based around particular software.[24] Pupils can put into practice what they know about using technology safely, respectfully and responsibly, as well as developing these skills in an immediately meaningful context.

Communities like these feature prominently in the work of computing professionals, many of whom are generous in sharing their work, expertise and experience.[25]

Inclusion

The digital divide

It is important to help pupils realise that access to technology can bring benefits and power, but that not everyone has easy access. Lack of access to technology can disadvantage particular groups or individuals within society.

Think carefully about whether any groups of pupils are excluded from, or disadvantaged by, activities you plan. For example, basing lessons on mobile phone apps or computer games may disadvantage those without access to such technology at home; providing resources or activities for pupils to access online from home seems unfair to those without internet access at home. Introducing lunchtime and after-school clubs is a practical way of making access available to all. If funds allow, consider providing pupils without their own computer with an old school computer that they can use at home.

Gender and inclusion

It is important to counter the stereotypes often associated with information technology and computing (e.g. that it is a male-only field). Efforts should be made, for example, in the selection of historical or contemporary case studies, to reflect the positive contributions of female practitioners such as Ada Lovelace, Grace Hopper or Dame Wendy Hall. Project topics should also be carefully considered to appeal to both genders.

Assistive technology

As with other areas of the curriculum, computing can be made more accessible to pupils with special educational needs or disabilities through the use of assistive technology – from adapted mice or keyboards, to screen readers and Braille displays. Within the curriculum, pupils might evaluate whether software and digital content, including those they create themselves, are accessible to users with special needs. At key stage 2, pupils might learn about assistive technology as examples of 'forms of input and output'.

English as an additional language

Technology can also facilitate the inclusion of pupils learning English as an additional language. The user interface of the operating system or application software can be set to languages other than English. Scratch and Snap! programs, for example, can be written in a variety of languages. Machine translation may also be useful for project work in which pupils learn about the opportunities offered by the internet.[26]

[23] See http://simple.wikipedia.org/ for the Simple English version of Wikipedia.
[24] See, for example, http://scratch.mit.edu/discuss/
[25] See, for example, https://github.com/ and http://stackoverflow.com/
[26] This section on inclusion is based on the Naace/CAS joint guidance: http://naacecasjointguidance.wikispaces.com/Terminology

Gifted and talented pupils

There are many opportunities for enrichment in computing, which need not be limited to talented or gifted pupils. There are perhaps parallels with music education, where it is not uncommon for primary pupils to be accomplished musicians in their own right, through independent study outside of school. The school can support and encourage by celebrating achievements and providing opportunities for pupils to pursue their interests.

There is a range of possibilities for independent learning, perhaps using resources or online communities to provide stimulus or support beyond what your school can offer. Your role might encompass steering very able pupils towards the best resources, providing critical feedback on their work, or setting further challenges.

Look for ways to enrich pupils' experience of computing rather than accelerating them through the syllabus. The provisional nature of work on computers allows scope for work to be refined and developed. Encouraging pupils to think about the algorithms and programs of applications they use is an effective way to develop some aspects of computational thinking, for instance by asking them to predict what will happen when they adopt a particular strategy in a computer game, or to consider how an image file changes when the brightness or colour is adjusted.

You can also provide, or allow pupils to choose, different sets of tools. For instance, programming tasks accomplished by most pupils in Scratch could be tackled in Logo or Python by particularly advanced pupils, or they might use Adobe Premiere Elements for video editing undertaken in Movie Maker by the rest of the class.

Many schools have implemented successful 'Digital Leaders' schemes, in which some pupils take responsibility for aspects of technology in the classroom or school. Although talented or gifted pupils can be a useful source of technical support or peer mentoring, it's important to ensure that they too are making progress.

Informal learning

There is scope for pupils to learn more about computing for themselves outside of school, and it would be good to encourage and celebrate this in school.

Many of the resources suitable for teaching computing in school are available free for pupils to use at home if they have a computer of their own.

Many schools have set up Code Clubs, often with external support, perhaps through someone working in the information technology industry. Code Club[27] make available carefully constructed resources and plans, and help manage DBS clearance for volunteers wanting to help schools in this way.

Other face-to-face events, such as Raspberry Jams, Young Rewired State and CoderDojo, are perhaps more suitable for secondary pupils. However, they have no lower age limit, although some parental involvement would be expected.

Scratch and Kodu have vibrant online forums, with ample opportunity for primary pupils to learn from others and to share their expertise as part of a moderated, global community.

[27] See www.codeclub.org.uk

Assessment

Formative assessment

There are certainly some challenges to assessing computing.

- It's hard for teachers to judge pupils' knowledge and understanding based on the outcomes of practical tasks alone.
- If pupils work collaboratively, it can be hard to identify each individual's contribution.
- If the teaching of computing is embedded in other subjects, it's often difficult to separate attainment in computing from that in the host subject.

Despite these challenges, the assessment for learning (AfL) techniques that you're familiar with in other subjects still apply. Let's look at some of the AfL approaches and consider how they can be applied to computing.

- **Self-assessment:** The curriculum expects pupils to debug their own programs, use logical reasoning to explain simple algorithms (including their own), and detect and correct errors in both algorithms and programs. One way to encourage self-assessment is for pupils to maintain a blog or video log of their work in computing, incorporating a reflective commentary alongside examples of what they've done.

- **Peer-assessment:** The ideas for self-assessment suggested above translate naturally into peer-assessment, with pupils working with a partner to review, and help correct, algorithms and programs, or providing critical, constructive feedback on digital content. Methods used by professional software developers, such as programming in pairs[28] and reviewing code, translate easily into the classroom. Online feedback and discussion, whether in the Scratch community or on pupils' blogs, also facilitate peer-based assessment.

- **Open questioning:** Pupils' knowledge of the concepts covered by the programme of study may not be immediately apparent in the work they produce. The use of open questioning is one way in which you can both assess and develop their grasp of concepts. 'Why' and 'how' questions work well: *Why did Google place that result at the top? How does your program work? Why might that not be a safe website?*, etc.

- **Discussion with peers:** Encouraging pupils to use similar open questions can be effective in allowing them to focus on what they've *learned*, rather than only on what they've *done*. Moving some of this discussion online, and perhaps involving pupils in other schools or countries, would be one powerful way to illustrate the opportunities offered by computer networks for communication and collaboration.

- **Target setting:** Project management skills such as planning, organising, motivating others and allocating resources, are of great importance in real-world projects, and they can be widely applied in education. The 'decomposition' aspect of computational thinking, in which large problems are broken down into small tasks, is a necessary part of managing all but the smallest of projects.

- **KWL:** Using lists to identify what pupils already ***know***, what they ***want*** to learn and subsequently what they have ***learned*** is a useful technique that can be used to support independent learning in computing. In particular, this can be applied to the logical reasoning needed to explain algorithms and to detect and correct errors, with pupils first establishing a firm foundation, before exploring alternatives and subsequently reviewing what they have learned, rather than only what they have done.

[28] **Williams, L. A. and Kessler, R. R., *All I Really Need to Know About Pair Programming I Learned in Kindergarten*, pp.108–114 (Communications of the ACM, 43(5) 2000).**

Using technologically enhanced learning can be particularly effective in assessment for learning, as some of the above suggestions indicate. Perhaps the most immediate opportunities are through the following.

- **Blogs:** There are now many examples of English primary pupils routinely recording and sharing their learning with a global audience through the use of class blogs. Individual pupil blogs can be a powerful tool to encourage self- and peer-assessment, track progress, give feedback, collate evidence, and share work with parents. It's now unnecessary to print off work from computing lessons when work can be attached to a reflective commentary on a pupil's blog, or saved to an area of the network or learning platform.

- **Automatic feedback:** A number of sites offer interactive tutorials in programming languages, providing immediate feedback on the success or failure of code in response to simple challenge questions. While few of these sites are aimed at primary school pupils, they may be of use for gifted or talented pupils eager to learn more programming independently.[29]

Summative assessment

National curriculum assessment has undergone considerable change for the new framework. The national curriculum review expert panel recommended that:

Attainment Targets in the presently established level descriptor form should not be retained.[30]

Responding to their report in June 2012, Michael Gove confirmed that:

In order to ensure that every child is expected to master this content, I have ... decided that the current system of levels and level descriptors should be removed and not replaced.[31]

So the attainment targets in all national curriculum subjects merely state:

By the end of each key stage, pupils are expected to know, apply and understand the matters, skills and processes specified in the relevant programme of study.[32]

This establishes a direct link between the contents of the programme of study and its assessment. Subsequent DfE guidance has made clear that:

Schools will be able to introduce their own approaches to formative assessment, to support pupil attainment and progression. The assessment framework should be built into the school curriculum, so that schools can check what pupils have learned and whether they are on track to meet expectations at the end of the key stage, and so that they can report regularly to parents.[33]

Perhaps the most obvious way to address this is to adopt an entirely criteria-based approach to assessment, with teachers forming a judgement as to whether each child has learned all the content of the programme of study by the end of the key stage.

The evidence to support this judgement can be assembled over the course of the key stage and need not be an onerous burden: as a child demonstrates their mastery of part of the curriculum the statement could be 'ticked off', with evidence of this achievement forming part of the child's computing portfolio or blog. It's likely that many pupils will assemble a lot of evidence for some statements and less for others, but some evidence of mastering each element should suffice to demonstrate meeting the expectations of the attainment targets.

Moreover, as a pupil's profile of achievement is built up, the statements yet to be achieved should provide a clear guide for planning, showing exactly where the 'gaps' are in each pupil's knowledge, skills and understanding, and thus where subsequent teaching should be targeted.

[29] See, for example, www.khanacademy.org/cs/programming, www.codeavengers.com/#learner or www.tryhaskell.org

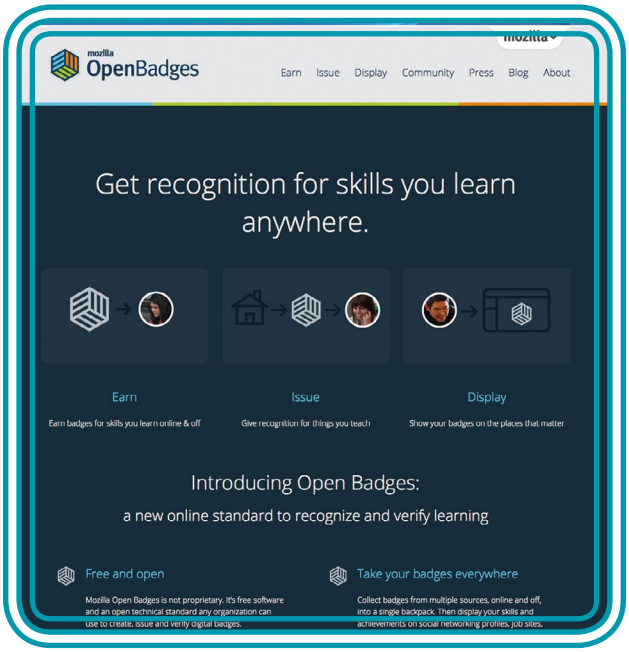
[30] See www.gov.uk/government/uploads/system/uploads/attachment_data/file/175439/NCR-Expert_Panel_Report.pdf (p.9).

[31] See http://media.education.gov.uk/assets/files/pdf/l/secretary%20of%20state%20letter%20to%20tim%20oates%20regarding%20the%20national%20curriculum%20review%2011%20june%202012.pdf (p.3).

[32] See www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study

[33] See www.education.gov.uk/schools/teachingandlearning/curriculum/nationalcurriculum2014/a00225864/assessingwithout-levels

Comparing individual profiles, and the evidence on which they're based, at the beginning and end of the year, should provide ample evidence of progress, of a far more meaningful nature than 'two sub-levels', specifying exactly what has been learned that year that wasn't already known. For this to work effectively, it might be sensible to break down the programme of study statements into their constituent clauses. As pupils achieve individual clauses, or perhaps as they achieve all the components of a statement from the programme of study, their achievement could be recognised through some form of badge. Mozilla's OpenBadges system[34] provides one possible solution.



While the DfE and others make a strong case for the abolition of attainment levels, their use is ingrained in many teachers' professional practice, as well as in the systems schools have in place to monitor pupils' progress. Nothing in the DfE's guidance prevents schools from continuing to use levels to monitor progress, and it seems likely that some schools will choose to do so, at least for the short to medium term.

In developing its computer science curriculum, Computing at School produced a set of level descriptors[35] for computer science, which might be used, perhaps with a little modification, alongside some statements from the old ICT attainment targets[36] to report progression on the computer science (CS), information technology (IT) and digital literacy (DL) components of the programme of study.

Alternatively, bearing in mind the emphasis on a direct link between what's taught and what's assessed, it's possible to take the statements from the programme of study and arrange them into some sort of order, from easier to harder statements. A somewhat arbitrary numbering might suggest a structure similar to the levels of the old attainment targets. For example, see the table on the next page. Note that this table is meant for illustration only, without any implication that these stages equate to old levels.

Another approach to levelling, although perhaps not in the spirit of the DfE's guidance, is to look at the nature of activities and the capabilities demonstrated by pupils separated from the subject content itself, perhaps using Bloom's revised taxonomy, or something similar, as a guide.

- Remembering
- Understanding
- Applying
- Analysing
- Evaluating
- Creating

[34] **See www.openbadges.org/**
[35] **See www.computingatschool.org.uk/data/uploads/ComputingCurric.pdf (pp.21–22).**
[36] **See http://webarchive.nationalarchives.gov.uk/20110813032310/http://qcda.gov.uk/libraryAssets/media/Level_Descriptions_-_ICT.pdf for the most recent proposed revision.**

	CS	IT	DL
1	Understand what algorithms are Create simple programs	Use technology purposefully to create digital content Use technology purposefully to store digital content Use technology purposefully to retrieve digital content	Use technology safely Keep personal information private Recognise common uses of information technology beyond school
2	Understand that algorithms are implemented as programs on digital devices Understand that programs execute by following precise and unambiguous instructions Debug simple programs Use logical reasoning to predict the behaviour of simple programs	Use technology purposefully to organise digital content Use technology purposefully to manipulate digital content	Use technology respectfully Identify where to go for help and support when they have concerns about content or contact on the internet or other online technologies
3	Write programs that accomplish specific goals Use sequence in programs Work with various forms of input Work with various forms of output	Use search technologies effectively Use a variety of software to accomplish given goals Collect information Design and create content Present information	Use technology responsibly Identify a range of ways to report concerns about contact
4	Design programs that accomplish specific goals Design and create programs Debug programs that accomplish specific goals Use repetition in programs Control or simulate physical systems Use logical reasoning to detect and correct errors in programs Understand how computer networks can provide multiple services, such as the World Wide Web Appreciate how search results are selected	Select a variety of software to accomplish given goals Select, use and combine internet services Analyse information Evaluate information Collect data Present data	Understand the opportunities computer networks offer for communication Identify a range of ways to report concerns about content Recognise acceptable/unacceptable behaviour
5	Solve problems by decomposing them into smaller parts Use selection in programs Work with variables Use logical reasoning to explain how some simple algorithms work Use logical reasoning to detect and correct errors in algorithms Understand computer networks, including the internet Appreciate how search results are ranked	Combine a variety of software to accomplish given goals Select, use and combine software on a range of digital devices Analyse data Evaluate data Design and create systems	Understand the opportunities computer networks offer for collaboration Be discerning in evaluating digital content

Concluding remarks

This is a really exciting time to be a pupil at primary school. The opportunities that advances in technology will bring to your pupils as they grow up are hard to imagine. The curiosity, creativity and courage that you nurture in them now should endure as they move on through education and into adult life. To exploit fully the opportunities that current and future technology offers them, pupils will draw on the understanding of computing you provide them with, as well as confidence gained through working on a range of meaningful projects throughout their primary education.

It's a really exciting time to be a primary school teacher, too. Don't be daunted by the changes in the move from ICT to computing. Rather, see this as an opportunity to develop your own knowledge about computing and to learn to program, if you've never had the chance before. Although this might sound like hard work, it's actually great fun. You'll find that you make better use of the technology you have at home and in school, and also that you start to think a bit differently, looking at systems and problems in the same way a computer scientist does.

Glossary

algorithm – an unambiguous procedure or precise step-by-step guide to solve a problem or achieve a particular objective.

computer networks – the computers and the connecting hardware (wifi access points, cables, fibres, switches and routers) that make it possible to transfer data using an agreed method ('protocol').

control – using computers to move or otherwise change 'physical' systems. The computer can be hidden inside the system or connected to it.

data – a structured set of numbers, representing digitised text, images, sound or video, which can be processed or transmitted by a computer.

debug – to detect and correct the errors in a computer program.

digital content – any media created, edited or viewed on a computer, such as text (including the hypertext of a web page), images, sound, video (including animation), or virtual environments, and combinations of these (i.e. multimedia).

information – the meaning or interpretation given to a set of data by its users, or which results from data being processed.

input – data provided to a computer system, such as via a keyboard, mouse, microphone, camera or physical sensors.

internet – the global collection of computer networks and their connections, all using shared protocols (TCP/IP) to communicate.

logical reasoning – a systematic approach to solving problems or deducing information using a set of universally applicable and totally reliable rules.

output – the information produced by a computer system for its user, typically on a screen, through speakers or on a printer, but possibly though the control of motors in physical systems.

program – a stored set of instructions encoded in a language understood by the computer that does some form of computation, processing input and/or stored data to generate output.

repetition – a programming construct in which one or more instructions are repeated, perhaps a certain number of times, until a condition is satisfied or until the program is stopped.

search – to identify data that satisfies one or more conditions, such as web pages containing supplied keywords, or files on a computer with certain properties.

selection – a programming construct in which the instructions that are executed are determined by whether a particular condition is met.

sequence – to place programming instructions in order, with each executed one after the other.

services – programs running on computers, typically those connected to the internet, which provide functionality in response to requests; for example, to transmit a web page, deliver an email or allow a text, voice or video conversation.

simulation – using a computer to model the state and behaviour of real-world (or imaginary) systems, including physical and social systems; an integral part of most computer games.

software – computer programs, including both application software (such as office programs, web browsers, media editors and games) and the computer operating system. The term also applies to 'apps' running on mobile devices and to web-based services.

variables – a way in which computer programs can store, retrieve or change simple data, such as a score, the time left, or the user's name.

World Wide Web – a service provided by computers connected to the internet (web servers), in which pages of hypertext (web pages) are transmitted to users; the pages typically include links to other web pages and may be generated by programs automatically.[37]

[37] **Phil Bagge provides a useful glossary with more detailed explanations of some of these terms: see http://code-it.co.uk/csvocab.html**

Resources

Background

Computing at School Working Group, *Computer Science: A Curriculum for Schools* (Cambridge, 2012), available at: www.computingatschool.org.uk/data/uploads/ComputingCurric.pdf

The Royal Society, *Shut Down or Restart? The Way Forward for Computing in UK Schools* (London, 2012), available at: http://royalsociety.org/uploadedFiles/Royal_Society_Content/education/policy/computing-in-schools/2012-01-12-computing-in-Schools.pdf

Rushkoff, D., *Program or be Programmed: Ten Commands for a Digital Age* (OR Books, 2009).

Teaching Agency, *Subject Knowledge Requirements for Entry into Computer Science Teacher Training* (London, 2012), available at: http://academy.bcs.org/sites/academy.bcs.org/files/subject%20knowledge%20requirements%20for%20entry%20into%20cs%20teacher%20training.pdf

Subject knowledge

Armoni, M. and Ben-Ari, M., *Computer Science Concepts in Scratch* (Michal Armoni and Moti Ben-Ari, 2013).

Bentley, P.J., *Digitized: The Science of Computers and How it Shapes our World* (Oxford University Press, 2012).

Berners-Lee, T., *Answers for Young People*, available at: www.w3.org/People/Berners-Lee/Kids.html

Blum, A., *Tubes: Behind the Scenes at the Internet* (Penguin, 2013).

Brennan, K. and Resnick, M., 'New frameworks for studying and assessing the development of computational thinking' (2012), available at: http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Computing at School, *The Raspberry Pi Education Manual* (CAS, 2012), available at: http://pi.cs.man.ac.uk/download/Raspberry_Pi_Education_Manual.pdf

Papert, S., *Mindstorms: Children, Computers, and Powerful Ideas* (Basic Books, 1993).

Petzold, C., *Code: The Hidden Language of Computer Hardware and Software* (Microsoft Press, 2009).

Teaching resources and ideas

Code Club provides detailed plans and resources for extra-curricular clubs, which might be adapted for use within the school curriculum. Free registration required: see www.codeclub.org.uk

New Zealand-based **Computer Science (CS) Unplugged** produce an excellent collection of resources exploring computer science ideas through classroom-based, rather than computer-based, activities: see http://csunplugged.org/

Computing at School (CAS) hosts a large resource bank of plans, resources and activities. CAS is free to join: see www.computingatschool.org.uk

CAS Primary Master Teachers; for example, one teacher has shared detailed lesson plans for computer science and digital literacy topics via his website at www.code-it.co.uk

CAS has made available a large collection of lesson plans and other resources through the **Digital Schoolhouse** project, based at Langley Grammar School: see www.digitalschoolhouse.org.uk

Naace (the ICT association) and **CAS** have developed joint guidance on the new computing curriculum: see http://naacecasjointguidance.wikispaces.com/home

A group of teachers and teacher trainers convened by the **NCTL** worked together to curate resources for initial teacher training for the computing curriculum, many of which may be useful for CPD and classroom use: see http://bit.ly/ittcomp

There are excellent resources available for teaching with MIT's Scratch programming toolkit, together with an online support community, on the **ScratchEd** site: see http://scratched.media.mit.edu/

Resources for teaching **safe, respectful and responsible use of technology** are widely available. Good starting points for exploring these topics are www.childnet.com/teachers-and-professionals and https://www.thinkuknow.co.uk/teachers/

Media

Mainstream television often broadcasts programmes relevant to topics in computing, and YouTube has a range of material, from video tutorials to academic lectures.

BBC Learning produced a collection of clips relating to computing in real-world contexts, and companion pieces exploring these in classroom contexts: see www.bbc.co.uk/programmes/b01r9tww/clips

The 2008 **Royal Institution Christmas Lectures** were given by computer scientist Chris Bishop. These can be watched at www.richannel.org/christmas-lectures/2008/2008-chris-bishop

TED has many high-quality 20-minute talks on computing topics that would be accessible to primary school pupils: see www.ted.com/topics/technology and http://ed.ted.com/lessons?category=technology for a curated collection of videos for schools.

The first two episodes of the BBC's **Virtual Revolution** are available online. These provide some excellent background material on the internet and the web: see www.bbc.co.uk/virtualrevolution/archive.shtml

Compared to ten years ago, there is now a wealth of programming environments designed specifically for primary schools. You may well have heard of Logo, Scratch and Kodu, but there many others, each with a different flavour and focus. You can find a growing list on the Computing At School website (www.computingatschool.org.uk/primary). Remember – programming at primary is now well-supported, engaging and fun!

Support

Computing at School (CAS), as the subject association for computer science, has been a key influence on the development of the new computing curriculum. CAS has a vibrant support community, including members from industry and from all phases of education. There's a dedicated forum for members in primary education, and many local and regional events. See www.computingatschool.org.uk for more information or to join (free membership).

Naace is the ICT association concerned with advancing education through the use of technology, both within and beyond the computing curriculum. Naace members share a vision for the role of technology in transforming learning and teaching. Its members include teachers, school leaders, advisors and consultants working within and across all phases of UK education. Membership requires an annual subscription but many resources are available free: see www.naace.co.uk

CAS has worked in collaboration with the British Computer Society (BCS) to establish a **Network of Teaching Excellence in Computer Science**. The network coordinates and provides training opportunities for serving and trainee teachers. The initiative is supported by the DfE, OCR (examination board), CPHC (Council of Professors and Heads of Computing), Microsoft and Google. The programme aims to build a high-quality, sustainable CPD infrastructure at low cost by nurturing long-term collaboration between employers, universities, professional bodies, schools and teachers: see www.computingatschool.org.uk/index.php?id=noe

Many local authorities and **CLCs (City Learning Centres)** provide support and advice for schools and teachers on all aspects of the curriculum, including computing. Contact your local advisors or consultants for details of events and support in your area.

Twitter is a great informal source of ideas and advice once you've built up a useful list of contacts. The CAS Twitter account: **@compatsch**, its followers: https://twitter.com/CompAtSch/followers and those it follows: https://twitter.com/CompAtSch/following may be helpful in developing your own personal learning network.

Background

Looking back at the last thirty years or so of computers in primary schools, there are two quite distinct threads: learning *about* computers and learning *with* computers. While this publication and the computing programme of study are concerned with the former, the latter has a crucial role in teaching and learning in the third millennium.

In the earliest days of BBC Micros in primary schools, creative programmers (many of them teachers) developed highly engaging educational software, from simple programs to practise arithmetic and spelling, through simulations and rudimentary virtual worlds, to tools to think with such as Logo. At the same time, a growing number of pupils were being bought home computers, mainly as games consoles, and dabbling with typing in and debugging (correcting) lines of code.

While programming or 'control' was an intrinsic part of the first national curriculum (1990), in which Information Technology Capability formed part of 'Technology' as a subject, there was already reference to using software applications for tasks such as creating databases, word-processing, presenting work and modelling.

In most schools, for much of the following two decades, ICT (as the subject became known in 1999) came to be seen as developing pupils' skills with a set of office-productivity programs, or their educational equivalent. This provided much scope for creative work, some grasp of how information can be structured and some good problem-solving activities, but arguably little insight into computer science.

In recent years, many primary educators have favoured an 'embedded' approach to ICT, in which ICT capability could be developed through using computers in the meaning-rich contexts of other subjects. In their 2008 and 2011 reports, Ofsted reported positively on the quality of teaching and achievement in ICT in primary schools in general, but warned of weaknesses in some aspects of the ICT curriculum, such as control and data handling. Ofsted did, however, highlight positive examples of primary practice, such as game design projects using Scratch.

The 'Next Gen' report, commissioned by the Department of Culture, Media and Sport on the state of the UK games and visual effects industries, recommended that computer science be brought into the national curriculum as an essential discipline. Furthermore, in his speech at the Edinburgh Television Festival in 2011, Google's executive chairman Eric Schmidt described himself as 'flabbergasted' that computer science wasn't taught as standard, and that England thus risked throwing away its great 'computing heritage'.

BBC Micro Computer, c. 1980s. Copyright Science and Society Picture Library, Getty Images. Editorial #90766368.

The Royal Society was commissioned by the UK computing community to investigate the state of computing education in schools, publishing their *Shut Down or Restart?* report in January 2012. Their recommendations included a rebranding of ICT, suggesting a possible split of the subject into digital literacy, information technology and computer science, and proposing 'computing' as an umbrella term for the subject as a whole.

With these concerns in mind, the Secretary of State for Education announced at the 2012 BETT Show that he would 'disapply' the old programme of study and attainment targets for ICT from September 2012, allowing schools to develop their own schemes of work, and giving them the opportunity to teach programming and other aspects of computer science. Responding to the consultation on disapplication, the Secretary of State announced that ICT was to continue as a national curriculum subject with a new programme of study.

Subsequently the DfE announced that the British Computer Society and the Royal Academy of Engineering would coordinate the drafting of this new programme of study, drawing on stakeholders from computing and education. This draft was subsequently revised by the DfE, with the subject name changing from ICT to computing. There were further revisions after public consultations, with the final version published in September 2013, ready to take effect in all maintained schools in September 2014.

Although the change of name, from ICT to computing, does reflect a change in emphasis, it's important to remember that there's more to computing than computer science, and that there's more to computer science than programming. Much that we've taught in the past in ICT will fit within the information technology and digital literacy aspects of the computing curriculum, and schools that have taught the 'sequencing instructions' aspects of the old programme of study will be able to build on this foundation as they address the new computer science content.

COMPUTING AT SCHOOL
EDUCATE · ENGAGE · ENCOURAGE
In collaboration with BCS, The Chartered Institute for IT

Computing At School promotes the teaching of computing in schools. Our aim is to support all teachers and all schools, and to develop excellence in the teaching of computing in their classrooms. We provide resources, training, local conferences and workshops, regional hub meetings, online community forums and so much more! **Computing At School** is free to join. Sign up and find out about events in your area by visiting us at **www.computingatschool.org.uk/primary**.

Naace

Naace promotes the appropriate use of computing to support learning, teaching and school organisation. Our aim is to support and challenge all teachers and schools and also those who provide services to schools. Naace has existed as an advocate in this area for 30 years and makes a small charge for annual membership.

Visit **www.naace.co.uk/membership** to join and to find out more about the ICT Quality Mark and Third Millennium Learning Award.

An eBook version of this guide, which can be freely shared with colleagues, is available at:
www.computingatschool.org.uk/primary

ISBN 978-1-78339-143-1