COMPUTING

The Curriculum, Resources & Vocabulary

January 2020

CONTENTS

1 CURRICULUM

1.1 OVERVIEW

Computing combines three related but different subjects: *computer science*, *digital literacy* and *information technology*. Computer science is the scientific and practical approach to computation and its applications. Digital literacy is the knowledge, skills and behaviours used in range of digital devices which are seen as a network. Information technology is the application of computers to store, present, manipulate and transmit data.

The new curriculum has much more to learn about computer science than before. These tables contain the basic subject areas of computer science that Key Stage 1 and Key Stage 2 children (and teachers) should know about. There is a glossary at the end of this guide with much more detail on each area.

Subject area	Key Stage 1 National Curriculum
Algorithms	Algorithms are sets of general steps for computers or people which can be repeated and subdivided.
Programs	Programs are sets of algorithms for a computer to blindly follow to produce outputs from inputs.
Data	Computers store and communicate many different kinds of information, and make decisions based on binary data.
Computers	Many devices contain computers, which are electronic machines with parts for different functions. These parts are for inputs, outputs and storing data.
Networks	The web has lots of information we can see in web browsers. Web pages can contain different media and are joined with hyperlinks.

Subject area	Key Stage 2 National Curriculum
Algorithms	An algorithm is a general procedure that can be represented as a flowchart or in readable languages, which can include decisions, repetition and nested parts. They are planned, tested and corrected in parts.
Programs	A program is an unambiguous collection of statements for a specific purpose; with mechanisms for a computer to execute to perform desired tasks.
Data	Information is binary data with meaning. In programs, information is stored in constants and variables; it can be organised into sortable and searchable tables. Tables can be used to answer questions.
Computers	Computers (such as cameras, smartphones and PCs) execute programs. Applications run within an operating system. Operating systems manage the relationship between applications and hardware; each hardware component is designed for a specific role.
Networks	The internet and the world wide web are different. The internet is many computers (connected with cables, phone networks and wifi) which supports the web, email and communications. The information on the web is stored on web servers and is provided through the internet.

1.2 EACH YEAR GROUP

We can separate the computing curriculum into six areas: computers, networks, using computers, Online safety, net searching and coding. Computers, networks and coding are part of *computer science*; Online safety and net searching are part of *digital literacy*, and using computers is *information technology*.

The general pattern is that there is more to learn each year, and that (for now at least) coding takes up a large part of the curriculum, especially in Years 2, 5 and 6. Some year groups do not have any new work for some areas of the curriculum.

1.2.1 *Foundation Stage*

Subject area	Curriculum statements
Computers	Begin to recognise common uses of information technology in the home and school environment
Using computers	With support, begin to use technology to create digital content
Online safety	Find out where to go for help and support when they have concerns about images and pop-ups
Coding	Follow and predict the behaviour of simple sets of instructions

1.2.2 *Year 1*

Subject area	Curriculum statements
Computers	Recognise common uses of information technology in the home and school environment
Using computers	Use technology purposefully to create digital content
	To begin to be able to touch type, and to know what the Shift, Return and Backspace keys do
Online safety	Understand where to go for help and support when they have concerns about images, pop-ups or contact on the internet
Coding	Predict the behaviour of simple programs
	Understand what algorithms are and how they are implemented on digital devices

1.2.3 *Year 2*

Subject area	Curriculum statements
Computers	Recognise common uses of information technology beyond school
Using computers	Use technology purposefully to create, organise, store, manipulate and retrieve digital content
	Be able to touch type with increasing speed and accuracy
	Compare the benefits of using different programs to create digital content
Online safety	Use technology safely, keep personal information private, know what to do if you see online bullying
Coding	Create simple programs
	Use logical reasoning to predict the behaviour of simple programs
	Debug simple programs by using logical reasoning to predict the actions instructed by the code
	Understand that programs execute by following precise and unambiguous instructions

1.2.4 *Year 3*

Subject area	Curriculum statements
Computers	Recognise familiar forms of input and output devices and how they are used
	Make efficient use of familiar forms of input and output devices
Networks	Understand that computer networks enable the sharing of data and information
	Understand that the internet is a large network of computers and that information can be shared between computers
	Understand the difference between the world wide web and the internet
Using computers	With support, select and use a variety of software to accomplish goals
Online safety	Use technology safely and respectfully, keeping personal information private and recognising acceptable and unacceptable behaviour
Net searching	Use safe search technologies and recognise that some sources are more reliable than others
Coding	Design, write and debug programs that control or simulate virtual events
	Use logical reasoning to explain how some simple algorithms work

1.2.5 *Year 4*

Subject area	Curriculum statements
Computers	Use other input devices such as cameras or sensors
	Select, use and combine a variety of software, systems and content that accomplish given goals
Networks	Understand what servers are and how they provide services to a network
Using computers	Understand the difference between data and information
	With support select, use and combine a variety of software on a range of digital devices to accomplish given goals
Online safety	Use technology responsibly and understand that communication online may be seen by others
	Understand where to go for help and support when they have concerns about content or contact on the internet
Net searching	Understand how results are selected and ranked by search engines
Coding	Decompose programs into smaller parts
	Use logical reasoning to detect and correct errors in algorithms and programs

1.2.6 *Year 5*

Subject area	Curriculum statements
Computers	Understand the purpose of common computer components and be familiar with their appearance, location and operation
	Understand that computers store data in binary, and how binary numbers work
Networks	Begin to use internet services to share and transfer data to a third party
Using computers	Independently select, use and combine a variety of software to design and create content for a given audience
Online safety	Understand the need to only select age-appropriate content
	Understand about privacy in online gaming, messaging services and social media
Net searching	Use filters in search technologies effectively and appreciate how results are selected and ranked
Coding	Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems
	Design, write and test simple programs that follow a sequence of instructions or allow a set of instructions to be repeated
	Design, write and test simple programs with opportunities for selection, where a particular result will happen based on actions or situations controlled by the user
	Use logical reasoning to explain how increasingly complex algorithms work to ensure a program's efficiency

1.2.7 *Year 6*

Subject area	Curriculum statements
Networks	Understand how computer networks enable computers to communicate and collaborate
	Use internet services within their own creations to share and transfer data to a third party
Using computers	Independently select, use and combine a variety of software to design and create content for a given audience, including collecting, analysing evaluating and presenting data and information
	Design and create a range of programs, systems and content for a given audience
Online safety	Use technology respectfully and responsibly, and know about how to manage privacy and bullying in gaming, messaging and social media
	Identify a range of ways to report concerns about content and contact in and out of school
Net searching	Use filters in search technologies effectively and be discerning when evaluating digital content
Coding	Learn to code using a real-life coding language to produce programs with real-world applications
	Solve problems by decomposing them into smaller parts
	Use variables, sequence, selection and repetition in programs and use them to explore real-world systems
	Use logical reasoning to explain how increasingly complex algorithms work, and to detect and correct errors in algorithms and programs efficiently

2

RESOURCES

COMPUTING IN THE NATIONAL CURRICULUM A comprehensive overview of primary computing, which is highly recommended reading for any teacher.
tinyurl.com/casprimaryguide

OUR ONLINE SAFETY PAGE Our school website's page on online safety, which is kept up to date with developments in internet use and social media.
stbonaventures.eschools.co.uk/web/online_safety/270471

SMARTIE THE PENGIUN Key Stage 1 online safety resources from Childnet. Follow the adventures of Smartie the Penguin as he learns how to be safe on the internet, covering: pop ups and in-app purchasing, inappropriate websites for older children and online bullying.
childnet.com/resources/smartie-the-penguin

THE SMART CREW Childnet illustrates five online safety SMART rules for lower Key Stage 2. They include a real-life SMART crew of young people, who guide the cartoon characters in their quest, and help them make safe online decisions.
bit.ly/smart-crew

BAREFOOT CAS Easy-to-use cross-curricular resources, not reliant on computers, so computing concepts can be incorporated throughout the curriculum. They also have excellent self-teach notes to deepen teachers' understanding of computing concepts and vocabulary.
barefootcas.org.uk

CS UNPLUGGED Computer science without a computer. Their free resources are used around the world. Their lessons are highly creative and exciting, and introduce students to Computational Thinking through concepts like binary numbers and algorithms.
www.csunplugged.org

SIMPLE ENGLISH WIKIPEDIA Wikipedia with shorter sentences and simpler words and grammar, for children and people who find it harder to learn or read. Their articles are great for children to learn computing concepts and vocabulary.
simple.wikipedia.org/wiki/computer_science

3

GLOSSARY

3.1 COMPUTERS

CENTRAL PROCESSING UNIT (CPU) The part of a computer which carries out the general-purpose instructions of *programs*, including arithmetic, logic, control and *input/output* operations.

GRAPHICS PROCESSING UNIT (GPU) The part of a computer which is designed to quickly produce images and animations such as in a *user interface*. Its electronic circuitry is far more efficient than a CPU at processing large blocks of visual data in parallel.

HARDWARE All the physical parts of a computer, as opposed to its *data* and *software*. Hardware components include *memory*, processors and *input/output* devices, usually all connected via a main circuit board called the motherboard.

INPUT/OUTPUT (I/O) The communication between a computer and the outside world, such as a human or another computer. Input devices include the mouse, keyboard, touchscreen, cameras, sensors (accelerometers, ambient light sensors), storage devices and *routers*. Output devices include screens, printers, speakers, storage devices and routers.

MEMORY Computer components which record and store *data*. Modern computers have fast, small and more expensive memory called random access memory (RAM) close to the CPU, and slower, cheaper memory called persistent storage further away.

RAM is used to store data temporarily so that programs can use it; it no longer stores data when it has no power. This is why some operating systems lose your work when you don't save it.

OPERATING SYSTEM (OS) *Software* which manages the relationship between *hardware* and application *programs*, and supports a *user interface* for users to control applications. Operating systems include OS X, Windows, Linux, Android and iOS.

SOFTWARE Sets of rules or instructions, but different from *algorithms* in that they are written in precise language a computer can understand. A CPU 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 between the ideas in the algorithm and the computer's machine code.

USER INTERFACE (UI) The part of the *program* and *operating system* that the user sees on the screen. A graphical user interface provides easier-to-use visual tools to control applications.

3.2 NETWORKS

CASCADING STYLE SHEETS (CSS) A language used to describe the presentation of a document written in HTML or other markup languages. CSS is designed to allow web designers to separate the content of web pages from their style (layout, fonts, colours).

HYPERTEXT MARKUP LANGUAGE (HTML) The language used to create most web pages. Web browsers can read HTML files and render them into web pages. HTML can refer web browsers to *cascading style sheets* to define the look and layout of the content on web pages.

HYPERLINK A reference to *data* that the reader can access to clicking or tapping on the displayed text or image. On the *World Wide Web*, hyperlinks connect *web pages*, files and other content.

INTERNET A vast network of computers connected via cables, phone networks and WiFi which supports many services including the *world wide web*, email and communications, all using shared protocols (TCP/IP) to communicate.

NETWORK A collection of connected computers which can exchange *data* using an agreed method (protocol). The computers can be connected with cables, or wirelessly, for example via WiFi or Bluetooth. A network in a small area like a home, a school or an office is called a local area network (LAN).

ROUTER A device which forwards *data* packets between different *networks* and other routers. Routers act like traffic directors, reading intended destinations for data packets and sending them on to the next part of their route.

SWITCH On a local network, a device which connects together *computers*, printers, *servers* and other devices, receiving and forwarding data between cables. They are different to *routers* because they don't deal with the entire route, only the next leg.

UNIFORM RESOURCE LOCATOR (URL) A type of web address which names a web resource (such as a web page), specifies its address and says how to get to it (a protocol). They can also be used to transfer files, send emails and access *databases*. Most *web browsers* display web page URLs in their address bar.

WEB BROWSER A *software* application for retrieving, presenting and traversing resources on the *world wide web*, and helping users to do so with a *user interface*. Web browsers include Chrome, Firefox, Internet Explorer, Opera and Safari.

WEB SERVER A device which stores, processes, receives and delivers web pages and other content to and from users, mainly for the *world wide web*. Users communicate with web servers via *web browsers*, which request content using the HyperText Transfer Protocol (HTTP).

This is why URLs begin with `http://`. The URL has to tell web browsers not only which page to find but also how to get it – by using the HTTP.

WEBSITE A set of related web pages usually provided from a single domain, hosted on at least one *web server*. Websites are accessible via a network such as the *internet* through a URL.

WORLD WIDE WEB (WWW) An information space where documents and other web resources are identified by URLs, interlinked by *hyperlinks*, and can be accessed via the *internet*.

3.3 DATA

ARRAY A *data* type composed of elements which are identified and selected by their key. Arrays are used in almost every *program* to make it easier to store a collection of related *variables* which are accessed by choosing their key when the program runs.

BINARY A number system which represents numeric values with two symbols, typically 0 and 1. Decimal numbers have place value in powers of 10 (1, 10, 100 etc.), whereas binary numbers have place value in powers of 2 (1, 2, 4, 8, 16 etc.). The binary number 10110 has a decimal value of $16 + 4 + 2 = 22$.

BIT The basic unit of *information*, symbolised as *b* or *bit*. It can have only one of two values, usually represented as a 0 or a 1.

BYTE A unit of *information* equal to eight *bits*, symbolised as *B*. Multiples of bytes can be symbolised with the standard unit prefixes *kilo-* and *mega-* etc., but there is confusion as to whether these prefixes mean powers of 1000, or powers of 1024 (2^{10}). For kB and MB the difference is small but it quickly becomes significant above that.

kilo means one thousand, mega means one million, giga means one billion, and tera means one trillion. There are prefixes for powers of 1024 (kibi, mebi and gibi) but few people use them.

DATA Any sequence of symbols which can convey meaning when interpreted. Data is not *information*: it is meaningless on its own. In computers, data is represented, stored and transferred using *binary*. Data structures, such as *arrays* and *objects*, can store data of different types, including numbers, images, strings of text and even other data structures.

DATABASE An organised collection of related *data*. A database management system is *software* that allows users to enter, store and retrieve data to allow *programs* to be run using it. Programs can use the data to make decisions for users to run complex systems.

INFORMATION *Data* with meaning. When data is structured and processed so that it can be interpreted, and decisions or conclusions can be made using it, it becomes information.

PIXEL The smallest controllable point in a digital image. Each pixel is represented by one or more *bits*. A one-bit-per-pixel image (1 bpp) has pixels of two colours: black or white; an 8 bpp image has 256 colours per pixel, and a 24 bpp ('Truecolor') image can combine 256 different shades of red, green and blue.

Eight bits can make 256 colours because an eight-digit binary number can represent any decimal number from 0 to 255.

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.

3.4 CODING

ACTION *Commands* which are run on an *object*, and cause it to change behaviour. Actions like `UP`, `DOWN` or `STOP` can move an object. Actions are often called methods.

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

Think of algorithms as being like recipes, with steps to follow in order. There can be several ways of cooking the same dish. Programs are collections of algorithms, like a recipe book.

ASSIGNMENT OPERATOR A type of *operator* that is used to set or change the value of a *variable*. An example is `SET TO` which changes a variable's value: `a SET TO 2` will change the value of the variable `a` to 2.

BLOCK *Commands* that are grouped together and are run when a condition is met or when an event occurs. One could have a `WHEN CLICKED` block: the commands in the block would be run when a mouse click occurs. In 2Code, commands in a block are given the same indentation and background colour to show they are part of the same block. In many programming languages, blocks of code are surrounded by curly brackets `{}`.

BUG A mistake in code that prevents the *program* from behaving in the way the coder intended.

COLLISION DETECTION Detecting when two *sprites* on the screen bump into each other. They are often used in a game to detect when a character hits a 'baddy'.

COMMAND A single instruction within a *program*. A program usually contains several commands. Sometimes commands are called statements.

CONCATENATION Adding text together: one could add two pieces of text `'The cat' + ' sat on the mat'` to create a single piece of text `'The cat sat on the mat'`.

CONDITION The trigger for a *command* to be run. For example, the trigger determines whether or not to run the `IF` or `ELSE` *block* in an if/else command or whether to keep repeating in a repeat until command. In `IF a = b THEN PRINT 'something' ELSE PRINT 'nothing'`, the condition is `a = b`.

CONDITIONAL OPERATOR An *operator* (represented by a symbol) which is interpreted as either `TRUE` or `FALSE` depending on the values either side of it. It is used as part of a *condition*. Examples are *equals* (as in `IF a = b`) which will be `TRUE` if the values of `a` and `b` are the same. Other examples are *not equals* (`!=`), *less than* (`<`) and *greater than* (`>`).

CONSOLE LOG An output window for a *program* that is used for *debugging*. It is usually a scrolling list of messages saying what the program is doing, or showing errors in the program.

DEBUGGER A tool that helps fix problems in code. Debuggers often have a *console log* and the ability to pause a *program*, step through code line by line and inspect *variables*. Often programmers spend as much time debugging as writing code.

Debugging is one of the most rewarding aspects of coding. It is an excellent way to promote resilience, collaboration and thinking logically.

EVENTS An occurrence that causes a *block* of code to be run. The event could be time related or could be some user *input* such as pressing a key or tapping the screen.

FUNCTIONS In coding, a named group of *commands* that can be run many times. To save repetition the commands can be put into a *function* and given a name. Calling the function (use its name) will run all the commands in that function. An example could be mixing: in a recipe, one might perform the function called mixing (which includes commands like picking up the spoon, holding the bowl and circling your hand) several times.

In computer science, a function describes what you want to happen to an input to produce an output. An algorithm is one particular way of doing that.

Many programming languages have things also called functions, which are named groups of commands.

IF/ELSE A *command* that tests a *condition*. If the condition is true then the commands inside the `IF` *block* are run. If the condition is not true and there is an `ELSE` block, then the commands inside the `ELSE` block are run.

LOGICAL OPERATORS *Operators* that are used for combining *conditions*, allowing for complex tests to be made. The most common logical operators are `AND` and `OR`. A condition using a logical operator could be `IF a > 0 AND a < 10 THEN PRINT 'a is between 1 and 9'`. They are beyond the primary school curriculum but have been included in the 2Code *Gorilla advanced debug challenges*.

MATHEMATICAL OPERATOR An *operator* which is a mathematical statement, e.g. $+$, $-$, $\times$ and $\div$. The mathematical operator $+$ in the code `a + 2` will evaluate to 4 if the *variable* `a` equals 2.

OBJECT An element in a *program* that can be created and manipulated using the object's *actions* or *properties*. In 2Code all the elements on the screen are objects.

OPERATOR A symbol such as +, = or AND that represents a process to apply to the objects on either side. For example, `a + b` or `IF a OR b = 2`. In 2Code there are four types of operators: *assignment operators*, *conditional operators*, *mathematical operators* and *logical operators*.

PROGRAM A stored set of instructions written in a language understood by a computer that performs a specific task when executed. A computer requires programs to work, and typically executes the program's instructions in a *central processing unit*.

PROPERTIES Qualities that are associated with an *object*. Examples include colour, speed or angle. Properties of an object can be changed in a similar way to *variables* by using *assignment operators*.

REPETITION When a *program* repeats a set of *commands*, either a set number of times or until a condition is met. In 2Code this could be done using `REPEAT`, `REPEAT UNTIL` or by using a timer.

SELECTION A decision *command*, where a *program* chooses a different outcome depending on a *condition*, such as `REPEAT UNTIL` or `IF/ELSE`.

SPRITE An element on the screen, usually an image. Sprites are often animated and can be set to move around the screen. They are often used to represent characters within a game.

VARIABLE Things used to keep track of the things that can change while a *program* is running, for example, reading the on/off state of a switch, counting the number of swipes before running a *command*, or changing the numbers in a timer. The user, the program or another variable can change a variable value. In 2Code, variables can be either numbers or text.

Variables are like boxes that the computer can use to store information. Each variable needs to have a name, which should help the coder remember what it is. The information inside the box is called the variable value.