Laleham Gap School

ICT & Computing

KS3 Curriculum Overview

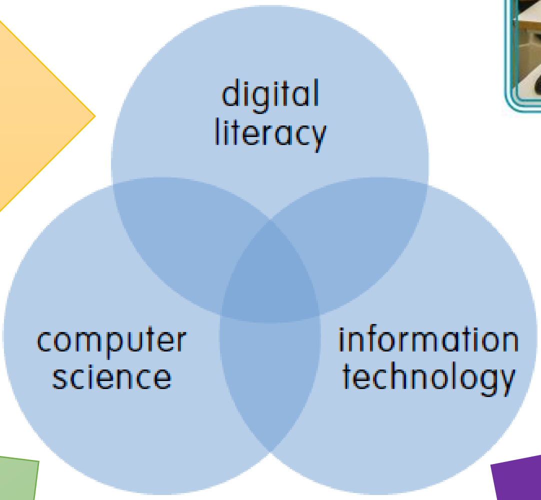
Computers are now part of everyday life and, for most of us, technology is essential to our lives, at home, school and at work. 'Computational thinking' is a skill that all pupils must learn if they are to be ready for the workplace and able to participate effectively in the digital world and in society as a whole.

The new national curriculum for computing has been developed to equip young people with the foundational skills, knowledge and understanding of computing they will need for the rest of their lives and it provides schools with an exciting new opportunity to reinvigorate teaching and learning in this important area of the curriculum.

Through the new programme of study for computing, pupils at Laleham Gap School will learn how computers and computer systems work, they will design and build programs, they will develop their ideas using technology, and create a range of digital content. Pupils will have 1 lesson per week in Years 7 & 8 and 2 lessons per week in Year 9.

The new curriculum identifies three distinct strands within computing, each of which is complementary to the others: Computer Science (CS), Information Technology (IT) and Digital Literacy (DL). Each component is essential in preparing pupils to thrive in an increasingly digital world.

Digital literacy is the ability to effectively, responsibly, safely and critically navigate, evaluate and create digital artefacts using a range of digital technologies.

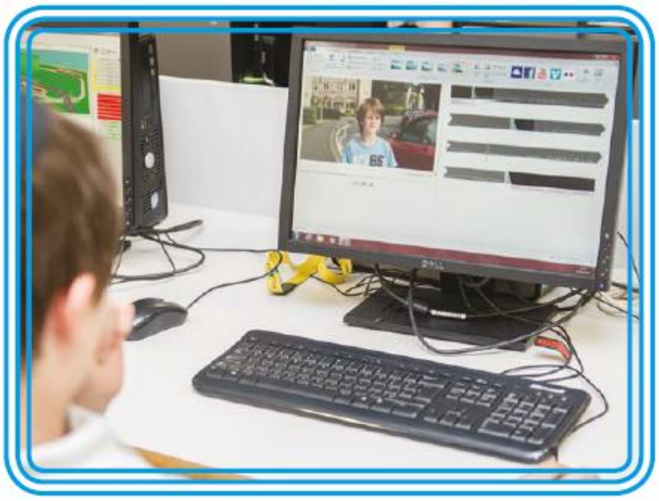


Computer science is the scientific and practical study of computation; what can be computed, how to compute it and how computation may be applied to the solution of problems.

Information technology is concerned with how computers and telecommunications equipment work, and how they may be applied to the storage, retrieval, transmission and manipulation of data.

Whilst participating in computing lessons, pupils at Laleham Gap School will experience all aspects of the curriculum and will benefit from a broad and engaging scheme of work which will take into account all 3 strands whilst satisfying the statutory requirements. The KS3 scheme of work will be based around the computing programmes of study, the subject content of which can be seen below. Pupils will be taught to:

	KS3
CS	Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems Understand several key algorithms that reflect computational thinking [for example, algorithms for sorting and searching]; use logical reasoning to compare the utility of alternative algorithms for the same problem Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures [for example, lists, tables or arrays]; design and develop modular programs that use procedures or functions Understand simple Boolean logic [for example, AND, OR and NOT] and some of its uses in circuits and programming; understand how numbers can be represented in binary, and be able to carry out simple operations on binary numbers [for example, binary addition, and conversion between binary and decimal] Understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems Understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits
IT	Undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users Create, re-use, revise and re-purpose digital artefacts for a given audience, with attention to trustworthiness, design and usability
DL	Understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct and know how to report concerns



Assessment of Computing at Laleham Gap

Formative assessment – Various techniques will be used to assess pupils learning in computing at Laleham Gap. These will include;

Self-assessment - Part of the process of becoming an independent learner is pupils being able to assess their own progress and evaluate their work. Self-assessment goes hand in hand with pupils setting their own goals.

Peer-assessment - Peer assessment provides discussion and feedback, helping the pupil and teacher to understand what a finished product would look like, and how to improve it. Pair programming and code reviews are techniques that can be used in the classroom.

Target setting - Setting challenging targets can help pupils recognise areas for development, an important step in becoming an independent learner. Targets will be realistic, manageable and fully evaluated.

Open questioning - Open questioning will allow pupils to understand the implications of theory. Pupil's work can be assessed by asking questions such as, "Why did you choose to do it this way and not another?" and "Can you explain how this works?"

Summative assessment - The levels from the previous ICT curriculum have been removed but 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.

The new programme of study will form the basis of Laleham Gap's assessment scheme. The school will aim to gather evidence from individual pupils as to whether they have met the requirements through a portfolio of their work. As a pupil demonstrates mastery of a particular point, the evidence will be collected and their progress recorded by the teacher in the school assessment system. The school will aim to use the new Progression Pathways Assessment Framework published by Computing at School, this framework provides guidance on what to look for at different stages in the development of knowledge under a range of subject headings. The expectations of the new computing curriculum have been mapped across to give a rough equivalence to the old National Curriculum Levels of the ICT Curriculum. This assessment framework is derived directly from the programme of study, organised as eight bands across KS1–KS3 for CS, IT and DL.

The Computing Progression Pathways can be roughly viewed as:

Pink & Yellow (KS1),	Equivalent to (Old NC Levels 1-2)
Orange & Blue (KS2),	Equivalent to (Old NC Levels 3-4)
Purple & Red (KS3),	Equivalent to (Old NC Levels 5-6)
Black & White (KS4),	Equivalent to (Old NC Levels 7-8).

An example of the assessment bands can be seen below.

Computing Progression Pathways

Pupil Progression	Algorithms	Programming & Development	Data & Data Representation
⬇	• Understands what an algorithm is and is able to express simple linear (non-branching) algorithms symbolically. (AL) • Understands that computers need precise instructions. (AL) • Demonstrates care and precision to avoid errors. (AL)	• Knows that users can develop their own programs, and can demonstrate this by creating a simple program in an environment that does not rely on text e.g. programmable robots etc. (AL) • Executes, checks and changes programs. (AL) • Understands that programs execute by following precise instructions. (AL)	• Recognises that digital content can be represented in many forms. (AB) (GE) • Distinguishes between some of these forms and can explain the different ways that they communicate information. (AB)
⬇	• Understands that algorithms are implemented on digital devices as programs.(AL) • Designs simple algorithms using loops, and selection i.e. if statements. (AL) • Uses logical reasoning to predict outcomes. (AL) • Detects and corrects errors i.e. debugging, in algorithms. (AL)	• Uses arithmetic operators, if statements, and loops, within programs. (AL) • Uses logical reasoning to predict the behaviour of programs. (AL) • Detects and corrects simple semantic errors i.e. debugging, in programs. (AL)	• Recognises different types of data: text, number. (AB) (GE) • Appreciates that programs can work with different types of data. (GE) • Recognises that data can be structured in tables to make it useful. (AB) (DE)
⬇	• Designs solutions (algorithms) that use repetition and two-way selection i.e. if, then and else. (AL) • Uses diagrams to express solutions. (AB) • Uses logical reasoning to predict outputs, showing an awareness of inputs. (AL)	• Creates programs that implement algorithms to achieve given goals. (AL) • Declares and assigns variables. (AB) • Uses post-tested loop e.g. 'until', and a sequence of selection statements in programs, including an if, then and else statement. (AL)	• Understands the difference between data and information. (AB) • Knows why sorting data in a flat file can improve searching for information. (EV) • Uses filters or can perform single criteria searches for information.(AL)
⬇	• Shows an awareness of tasks best completed by humans or computers. (EV) • Designs solutions by decomposing a problem and creates a sub-solution for each of these parts. (DE) (AL) (AB) • Recognises that different solutions exist for the same problem. (AL) (AB)	• Understands the difference between, and appropriately uses if and if, then and else statements. (AL) • Uses a variable and relational operators within a loop to govern termination. (AL) (GE) • Designs, writes and debugs modular programs using procedures. (AL) (DE) (AB) (GE) • Knows that a procedure can be used to hide the detail with sub-solution. (AL) (DE) (AB) (GE)	• Performs more complex searches for information e.g. using Boolean and relational operators. (AL) (GE) (EV) • Analyses and evaluates data and information, and recognises that poor quality data leads to unreliable results, and inaccurate conclusions. (AL) (EV)
⬇	• Understands that iteration is the repetition of a process such as a loop. (AL) • Recognises that different algorithms exist for the same problem. (AL) (GE) • Represents solutions using a structured notation. (AL) (AB) • Can identify similarities and differences in situations and can use these to solve problems (pattern recognition). (GE)	• Understands that programming bridges the gap between algorithmic solutions and computers. (AB) • Has practical experience of a high-level textual language, including using standard libraries when programming. (AB) (AL) • Uses a range of operators and expressions e.g. Boolean, and applies them in the context of program control. (AL) • Selects the appropriate data types. (AL) (AB)	• Knows that digital computers use binary to represent all data. (AB) • Understands how bit patterns represent numbers and images. (AB) • Knows that computers transfer data in binary. (AB) • Understands the relationship between binary and file size (uncompressed). (AB) • Defines data types: real numbers and Boolean. (AB) • Queries data on one table using a typical query language. (AB)

⬇	• Understands a recursive solution to a problem repeatedly applies the same solution to smaller instances of the problem. (AL) (GE) • Recognises that some problems share the same characteristics and use the same algorithm to solve both. (AL) (GE) • Understands the notion of performance for algorithms and appreciates that some algorithms have different performance characteristics for the same task. (AL) (EV)	• Uses nested selection statements. (AL) • Appreciates the need for, and writes, custom functions including use of parameters. (AL) (AB) • Knows the difference between, and uses appropriately, procedures and functions. (AL) (AB) • Understands and uses negation with operators. (AL) • Uses and manipulates one dimensional data structures. (AB) • Detects and corrects syntactical errors. (AL)	• Understands how numbers, images, sounds and character sets use the same bit patterns. (AB) (GE) • Performs simple operations using bit patterns e.g. binary addition. (AB) (AL) • Understands the relationship between resolution and colour depth, including the effect on file size. (AB) • Distinguishes between data used in a simple program (a variable) and the storage structure for that data. (AB)
⬇	• Recognises that the design of an algorithm is distinct from its expression in a programming language (which will depend on the programming constructs available). (AL) (AB) • Evaluates the effectiveness of algorithms and models for similar problems. (AL) (AB) (GE) • Recognises where information can be filtered out in generalizing problem solutions. (AL) (AB) (GE) • Uses logical reasoning to explain how an algorithm works. (AL) (AB) (DE) • Represents algorithms using structured language. (AL) (DE) (AB)	• Appreciates the effect of the scope of a variable e.g. a local variable can't be accessed from outside its function. (AB) (AL) • Understands and applies parameter passing. (AB) (GE) (DE) • Understands the difference between, and uses, both pre-tested e.g. 'while', and post-tested e.g. 'until' loops. (AL) • Applies a modular approach to error detection and correction. (AB) (DE) (GE)	• Knows the relationship between data representation and data quality. (AB) • Understands the relationship between binary and electrical circuits, including Boolean logic. (AB) • Understands how and why values are data typed in many different languages when manipulated within programs. (AB)
⇩	• Designs a solution to a problem that depends on solutions to smaller instances of the same problem (recursion). (AL) (DE) (AB) (GE) • Understands that some problems cannot be solved computationally. (AB) (GE)	• Designs and writes nested modular programs that enforce reusability utilising sub-routines wherever possible. (AL) (AB) (GE) (DE) • Understands the difference between 'While' loop and 'For' loop, which uses a loop counter. (AL) (AB) • Understands and uses two dimensional data structures. (AB) (DE)	• Performs operations using bit patterns e.g. conversion between binary and hexadecimal, binary subtraction etc. (AB) (AL) (GE) • Understands and can explain the need for data compression, and performs simple compression methods. (AL) (AB) • Knows what a relational database is, and understands the benefits of storing data in multiple tables. (AB) (GE) (DE)