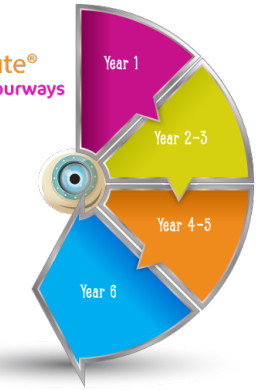
Computational Thinking Progression

Algorithms	Programming & Development	Data & Representation	Hardware & Processing	Communication & Networks	Information Technology
• 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 computers have no intelligence and that computers can do nothing unless a program is executed. **(AL)** • Recognises that all software executed on digital devices is programmed. **(AL) (AB) (GE)**	• Understands that computers have no intelligence and that computers can do nothing unless a program is executed. **(AL)** • Recognises that all software executed on digital devices is programmed. **(AL) (AB) (GE)**	• Obtains content from the world wide web using a web browser. **(AL)** • Understands the importance of communicating safely and respectfully online, and the need for keeping personal information private. **(EV)** • Knows what to do when concerned about content or being contacted. **(AL)**
• 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)	• Recognises that a range of digital devices can be considered a computer. **(AB) (GE)** • Recognises and can use a range of input and output devices. • Understands how programs specify the function of a general purpose computer. **(AB)**	• Navigates the web and can carry out simple web searches to collect digital content. **(AL) (EV)** • Demonstrates use of computers safely and responsibly, knowing a range of ways to report unacceptable content and contact when online. **(EV)**	• Uses technology with increasing independence to purposefully organise digital content. **(AB)** • Shows an awareness for the quality of digital content collected. **(EV)** • Uses a variety of software to manipulate and present digital content: data and information. **(AL)** • Shares their experiences of technology in school and beyond the classroom. **(GE) (EV)** • Talks about their work and makes improvements to solutions based on feedback received. **(EV)**

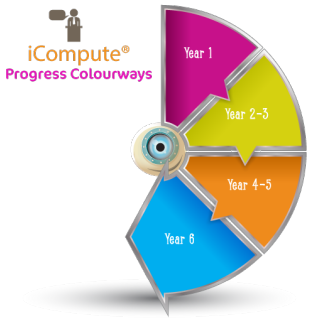
iCompute®
Progress Colourways



AL = Algorithms
DE = Decomposition
GE = Generalisation (Patterns)
AB = Abstraction
EV = Evaluation

Computational Thinking Progression

Algorithms	Programming & Development	Data & Representation	Hardware & Processing	Communication & Networks	Information Technology
• 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)**	• Knows that computers collect data from various input devices, including sensors and application software. **(AB)** • Understands the difference between hardware and application software, and their roles within a computer system. **(AB)**	• Understands the difference between the internet and internet service e.g. world wide web. **(AB)** • Shows an awareness of, and can use a range of internet services • Recognises what is acceptable and unacceptable behaviour when using technologies and online services. **(EV)**	• Collects, organises and presents data and information in digital content. **(AB)** • Creates digital content to achieve a given goal through combining software packages and internet services to communicate with a wider audience e.g. blogging. **(AL)** • Makes appropriate improvements to solutions based on feedback received, and can comment on the success of the solution. **(EV)**
• 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 why and when computers are used. **(EV)** • Understands the main functions of the operating system. **(DE) (AB)** • Knows the difference between physical, wireless and mobile networks. **(AB)**	• Understands how to effectively use search engines, and knows how search results are selected, including that search engines use 'web crawler programs'. **(AB) (GE) (EV)** • Selects, combines and uses internet services. **(EV)** • Demonstrates responsible use of technologies and online services,and knows a range of ways to report concerns.	• Makes judgements about digital content when evaluating and repurposing it for a given audience. **(EV) (GE)** • Recognises the audience when designing and creating digital content. **(EV)** • Understands the potential of information technology for collaboration when computers are networked. **(GE)** • Uses criteria to evaluate the quality of solutions, can identify improvements making some refinements to the solution, and future solutions. **(EV)**



AL = Algorithms
DE = Decomposition
GE = Generalisation (Patterns)
AB = Abstraction
EV = Evaluation

Assessment

Algorithmic thinking

Algorithmic thinking is the ability to think in terms of sequences and rules as a way of solving problems. It is a core skill that pupils develop when they learn to write their own computer programs. The following can be observed in the primary classroom.

The first set involves formulating instructions to achieve a desired effect.
Formulating instructions to be followed in a given order (sequence).

- Formulating instructions that use arithmetic and logical operations.
- Writing sequences of instructions that store, move and manipulate data (variables and assignment).
- Writing instructions that choose between different constituent instructions (selection).
- Writing instructions that repeat groups of constituent instructions (loops/iteration).
- Grouping and naming a collection of instructions that do a well-defined task to make a new instruction (subroutines, procedures, functions, methods).

It also involves:

- Using an appropriate notation to write code to represent any of the above.
- Creating algorithms to test a hypothesis.
- Creating algorithms that give experience-based solutions (heuristics).
- Creating algorithmic descriptions of real world processes so as to better understand them (computational modelling).

Computational Thinking Progression

Decomposition

Decomposition is a way of thinking about artefacts in terms of their parts. The parts can then be understood, solved, developed and evaluated separately.
The following can be observed in the classroom.

- Breaking down artefacts into constituent parts to make them easier to work with.
- Breaking down a problem into simpler versions of the same problem that can be solved in the same way (recursive and divide and conquer strategies).

Generalisation (Patterns)

Generalisation is a way of solving new problems based on previous problem solutions. It involves identifying and exploiting patterns.
The following behaviours can be observed in the classroom.

- Identifying patterns and commonalities in artefacts.
  Adapting solutions, or parts of solutions, so they apply to a whole class of similar problems.
- Transferring ideas and solutions from one problem area to another.

Abstraction

Abstraction is the process of making an artefact more understandable by hiding detail. The following behaviours can be observed in the classroom.

- Reducing complexity by removing unnecessary detail.
- Choosing a way to represent an artefact, to allow it to be manipulated in useful ways.
- Hiding the full complexity of an artefact (hiding functional complexity).
- Hiding complexity in data, for example by using data structures.
- Identifying relationships between abstractions.
- Filtering information when developing solutions.

Evaluation

Evaluation is the process of ensuring a solution is a good one: that it is fit for purpose. There is a specific and often extreme focus on attention to detail in computational thinking based evaluation.

The following behaviours can be observed in the classroom.

- Assessing that an artefact is fit for purpose.
- Assessing whether an artefact does the right thing (functional correctness).
- Designing and running test plans and interpreting the results (testing).
- Assessing whether the performance of an artefact is good enough (utility: effectiveness and efficiency).
- Comparing the performance of artefacts that do the same thing.
- Making trade-offs between conflicting demands
- Assessing whether an artefact is easy for people to use (usability).
- Assessing whether an artefact gives an appropriately positive experience when used (user experience).
- Assessment of any of the above against the specification and set criteria.
- Stepping through processes or algorithms/code step-by-step to work out what they do (dry run/tracing).
- Using rigorous argument to justify that an algorithm works (proof).
- Using rigorous argument to check the usability or performance of an artefact (analytical evaluation).
- Using methods involving observing an artefact in use to assess its usability (empirical evaluation).
- Assessing whether a product meets general performance criteria (heuristics).

Examples of algorithmic thinking, decomposition, generalisation, abstraction and evaluation are found across the computing curriculum and across the full range of attainment. Computational thinking is not age dependent and therefore the concepts are not attributed to years or key stages, but they are capability dependent.