A community of active learners who go above and beyond in everything they do, equipped to make a difference in their own lives and the lives of others.

# Computing

A high-quality computing education is the pen and paper of our time and is the lens through which we experience our world A high-quality computing education is the pen and paper of our time and is the lens through which we experience our world.

David Warlick

## Intent

We develop computational thinkers who can confidently communicate and solve problems, grounded in their own experiences. We cultivate an enthusiasm for digital literacy and enable all children to be members of, and contribute towards, a fast-changing dynamic world.

## National Curriculum Purpose of Study

A high-quality computing education equips children to use computational thinking and creativity to understand and change the world. Computing has deep links with mathematics, science, and design and technology, and provides insights into both natural and artificial systems. The core of computing is computer science, in which children 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, children are equipped to use information technology to create programs, systems and a range of content. Computing also ensures that children 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.

## Statement of Implementation

We use the Teach Computing curriculum, from the NCCE (National Centre for Computing Education), which uses the following pedagogical principles:

**Lead with concepts:** Support children in the acquisition of knowledge, through the use of key concepts, terms, and vocabulary, providing opportunities to build a shared and consistent understanding. Glossaries, concept maps, and displays, along with regular recall and revision, can support this approach.

**Unplug, unpack, repack:** Teach new concepts by first unpacking complex terms and ideas, exploring these ideas in unplugged and familiar contexts, then repacking this new understanding into the original concept. This approach (semantic waves) can help children develop a secure understanding of complex concepts.

Create projects: Use project-based learning activities to provide children with the opportunity to apply and consolidate their knowledge and understanding. Design is an important, often overlooked aspect of computing. Children can consider how to develop an artefact for a particular user or function and evaluate it against a set of criteria.

Challenge misconceptions: Use formative questioning to uncover misconceptions and adapt teaching to address them as they occur. Awareness of common misconceptions alongside discussion, concept mapping, peer instruction, or simple quizzes can help identify areas of confusion.

Structure lessons: Use supportive frameworks when planning lessons, such as PRIMM (Predict, Run, Investigate, Modify, Make) and Use-Modify-Create. These frameworks are based on research and ensure that differentiation can be built in at various stages of the lesson.

Work together: Encourage collaboration, specifically using pair programming and peer instruction, and also structured group tasks. Working together stimulates classroom dialogue, articulation of concepts, and development of shared understanding.

Model everything: Model processes or practices — everything from debugging code to binary number conversions — using techniques such as worked example s and live coding. Modelling is particularly beneficial to novices, providing scaffolding that can be gradually taken away.

Add variety: Provide activities with different levels of direction, scaffolding, and support that promote active learning, ranging from highly structured to more exploratory tasks. Adapting your instruction to suit different objectives will help keep all children engaged and encourage greater independence.

Make concrete: Bring abstract concepts to life with real-world, contextual examples and a focus on interdependencies with other curriculum subjects. This can be achieved through the use of unplugged activities, proposing analogies, storytelling around concepts, and finding examples of the concepts in children' lives.

Read and explore code first: When teaching programming, focus first on code 'reading' activities, before code writing. With both block-based and text-based programming, encourage children to review and interpret blocks of code. Research has shown that being able to read, trace, and explain code augments children' ability to write code.

Get hands-on: Use physical computing and making activities that offer tactile and sensory experiences to enhance learning. Combining electronics and programming with arts and crafts (especially through exploratory projects) provides children with a creative, engaging context to explore and apply computing concepts.

Foster program comprehension: Use a variety of activities to consolidate knowledge and understanding of the function and structure of programs, including debugging, tracing, and Parson's Problems. Regular comprehension activities will help secure understanding and build connections with new knowledge.

The Teach Computing journey is taught using the following concepts and skills, within a coherent and well-sequenced spiral curriculum. Each component provides all children with the knowledge and skills required to progress from one year group to the next.