Willoughby Primary School

Computing Curriculum Statement

As a school we aim to:

Deliver an engaging and inspirational curriculum that prepares children for the future, develops their **curiosity** and deepens their **understanding** of the world around them. We foster each child's **independence** to nurture character development, health and well-being.

In this subject we aim to:

In line with the National Curriculum for Computing, our aim is to provide a high-quality, inclusive computing education which allows children to use software and computational thinking to understand and thrive in an ever-changing technological world. Children's curiosity is nurtured through the curriculum, it teaches children key knowledge about how computers and computer systems work, and how they are designed and programmed. Learners will have the opportunity to gain an understanding of computational systems of all kinds, whether or not they include computers, by immersing themselves with a variety of software and hardware, all underpinned by the NCCE's 12 principles of Computing Pedagogy (Appendix 1).

Our children say:

They like computing as its fun and enjoyable. It's a useful thing to do and know how to do. They like typing up and publishing work on a computer. They do research on the i-pads and laptops in other subjects like Science, Geography and History. Both KS1 and KS2 say that they use I-pads, laptops, and bee bots.

Learning intentions:

By the time pupils leave Willoughby Primary School, they will have gained key knowledge and skills in the four main areas of computing:
Computer science (programming and understanding how digital systems work),
Computational thinking (solving problems through decomposition, pattern recognition, abstraction and algorithms)
Information technology (using computer systems to store, retrieve and send information)
Digital literacy (evaluating digital content and using technology safely and respectfully).
The objectives within each strand support the development of learning across the key stages, ensuring a solid grounding for love of learning and beyond.
At Willoughby we follow the scheme of work for computing from the NCCE Teach Computing website, which ensures relevant skills are taught throughout the year. Children focus on a topic each half term where they get to explore different software, practise and develop their skills in using it and then create their work using it. For topics that focus on understanding hardware, such as computing systems and networks, learning is planned to engagingly present this information and for children to creatively present their understandings.

Appendix 1

National Centre for Computing Education

Raspberry Pi

How we teach computing

12 pedagogy principles

Lead with concepts

Support pupils 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 pupils develop a secure understanding of complex concepts.

Create projects

Use project-based learning activities to provide pupils with the opportunity to apply and consolidate their knowledge and understanding. Design is an important, often overlooked aspect of computing. Pupils 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 examples 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 pupils 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 pupils' 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 pupils to review and interpret blocks of code. Research has shown that being able to read, trace, and explain code augments pupils' 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 pupils 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.

Find out more about our principles and add some or all to your personal pedagogy toolkit.