Computer Science

Curriculum Philosophy

The Computer Science Department at St. George's is committed to embedding and highlighting **Christian values** in our curriculum, specifically dignity, hope, community, wisdom, humility, and kindness. We **centre** our teaching on **recognising** the inherent value and worth of each individual, aiming to achieve this by providing a **high-quality computing education** that equips pupils to use **computational thinking** and creativity to understand and change the world.

To **instil hope** in our pupils, we explore technological advancements and pioneers who offer inspiration and challenge pupils' perspectives, addressing possible misconceptions about the digital world. We encourage thought-provoking discussions and **collaborative projects** to foster **personal growth** and a sense of **community** within the classroom.

Pupils reflect upon the values of **wisdom** and **humility** to broaden their perspectives and challenge their own beliefs about technology and its impact on society. We strive to develop **critical thinking** and **problem-solving skills**, deepening pupils' understanding of the **complexities of computer science**, its ethical implications, and its influence on human societies, cultures, and thought.

As a discipline, computer science inspires curiosity and fascination with how technology shapes our world and the endless possibilities it offers for the future. Our curriculum covers **foundational computing principles** as outlined in the national curriculum, ensuring pupils understand and apply the **fundamental principles and concepts of computer science**, including abstraction, logic, algorithms, and data representation. We aim to enhance pupils' **clarity of thought** and **self-expression** through the development of **powerful knowledge** and **technical vocabulary**. It is our intention that our pupils are not constrained by any **barriers**—social, cultural, or technological.

We challenge our pupils to **think as computer scientists**. Therefore, we have developed five departmental "**principles**" that the teaching team adheres to:

- **Direct Instruction and Modelling**: We are devoted to teaching **foundational computing concepts** through clear, direct instruction and modelling. By demonstrating processes and solutions, we equip pupils with the necessary schemas to understand and apply their knowledge effectively.
- **Develop Computational Thinking**: We focus on building pupils' abilities to **think algorithmically** and solve complex problems using abstraction, logic, algorithms, and data representation.
- **Facilitate Guided Practice and Application**: After modelling, we provide opportunities for pupils to apply their knowledge through **guided practice**, allowing them to solve problems using the base knowledge as a schema.
- **Contextualise Learning**: We support pupils in connecting **theoretical concepts to real-world applications** to develop a practical understanding of how digital systems work and how to put this knowledge to use through programming.
- **Utilise a Concept-Driven Curriculum**: We employ a curriculum **centred** on **key computing concepts** to build deep and transferable understanding, ensuring pupils can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems.

Curriculum Sequencing Rationale

In order to achieve an increasingly sophisticated understanding of computer science, topics have been intelligently sequenced based on the following rationale:

- **Foundation Building Through Direct Instruction**: We begin with foundational concepts such as understanding the principles of information and computation, how digital systems work, and basic programming skills. These are taught through direct instruction and modelling to ensure all pupils have a solid base upon which to build more complex ideas.
- **Progressive Complexity with Guided Practice**: Topics are arranged to gradually increase in complexity. Pupils apply their knowledge through **guided practice**, allowing them to make connections between concepts and develop problem-solving skills using their established schemas.
- **Interdisciplinary Links**: We incorporate elements of mathematics, science, and design and technology to provide a holistic understanding of how computer science interacts with other disciplines. Pupils learn to understand key algorithms, use logical reasoning, and apply mathematical concepts such as binary numbers and Boolean logic.

- **Real-World Applications**: We sequence topics to align with **current technological trends and societal needs**, making learning relevant and engaging. Pupils undertake projects that involve selecting, using, and combining multiple applications to achieve challenging goals, including collecting and **analysing** data and meeting the needs of known users.

Curriculum Building Blocks

To maximise the impact of our curriculum, the following elements have been selected as 'building blocks':

- **Direct Instruction and Modelling**: Clear teaching of concepts and procedures, providing pupils with the necessary knowledge and skills. Teachers model problem-solving processes, programming techniques, and computational methods.
- **Structured Practice and Application**: Pupils engage in structured activities that reinforce learning. **Guided practice** transitions to independent practice as pupils become more proficient.
- **Computational Thinking Skills**: Developing problem-solving strategies, logical reasoning, and algorithmic thinking. Pupils **analyse** problems in computational terms and gain practical experience in writing computer programs.
- **Understanding Digital Systems**: Knowledge of hardware and software components that make up computer systems, how they communicate, and how instructions are stored and executed.
- **Digital Literacy and Ethical Use**: Ensuring pupils become responsible, competent, confident, and creative users of information and communication technology. Pupils learn to use technology safely, respectfully, responsibly, and securely.

Addressing Social Disadvantage and Inclusivity

The computer science curriculum addresses social disadvantage by bridging gaps in pupils' knowledge and skills:

- **Equitable Access to Knowledge**: Through direct instruction, all pupils receive the same **high-quality teaching**, ensuring that everyone has access to the foundational knowledge required to succeed.
- **Supportive Learning Environment**: We provide additional support where needed, including resources and interventions for pupils who may require extra help in understanding key concepts.

- **Inclusivity Through Representation**: We highlight contributions from diverse cultures and communities within the field of computer science, promoting respect for diversity and encouraging all pupils to see themselves as potential contributors to the field.

Personal Development and Christian Values

We believe that computer science contributes to the personal development of pupils at St. George's:

- **Building Confidence Through Mastery**: By providing clear instruction and opportunities for success, pupils develop confidence in their abilities, reflecting the Christian value of hope.
- **Developing Resilience**: Pupils learn to persevere through challenging tasks, such as debugging code, which builds personal resilience.
- **Promoting Ethical Understanding**: Discussions on topics like data privacy and digital citizenship help pupils consider the moral implications of technology, fostering spiritual and moral development.
- **Encouraging Collaboration and Respect**: Group projects and peer learning foster teamwork, communication skills, and respect for others' ideas, enhancing social development and a sense of community.

The Computer Science Department at St. George's is committed to embedding and highlighting **Christian values** in our curriculum, specifically **dignity, hope, community, wisdom, humility, and kindness**. We centre our teaching on recognising the **inherent value and worth of each individual**, aiming to achieve this by providing a **high-quality computing education** that equips pupils to use **computational thinking** and **creativity** to understand and change the world.

To **instil hope** in our pupils, we explore technological advancements and pioneers who offer inspiration and challenge pupils' perspectives, addressing possible misconceptions about the digital world. We encourage thought-provoking discussions and **collaborative projects** to foster **personal growth** and a sense of **community** within the classroom.

Pupils reflect upon the values of **wisdom** and **humility** to broaden their perspectives and challenge their own beliefs about technology and its impact on society. We strive to develop **critical thinking** and **problem-solving skills**, deepening pupils' understanding of the **complexities of computer science**, its **ethical implications**, and its influence on human societies, cultures, and thought.

As a discipline, computer science inspires curiosity and fascination with how technology shapes our world and the endless possibilities it offers for the future. Our curriculum covers **foundational computing principles** as outlined in the national curriculum, ensuring pupils understand and apply the **fundamental principles and concepts of computer science**, including **abstraction, logic, algorithms, and data representation**. We aim to enhance pupils' **clarity of thought** and **self-expression** through the development of **powerful knowledge** and **technical vocabulary**. It is our intention that our pupils are not constrained by any **barriers**—social, cultural, or technological.

We challenge our pupils to **think as computer scientists**. Therefore, we have developed five departmental "**principles**" that the teaching team adheres to:

1. **Direct Instruction and Modelling:** We are devoted to teaching **foundational computing concepts** through clear, direct instruction and modelling. By demonstrating processes and solutions, we equip pupils with the necessary schemas to understand and apply their knowledge effectively.
2. **Develop Computational Thinking:** We focus on building pupils' abilities to **think algorithmically** and solve complex problems using abstraction, logic, algorithms, and data representation.
3. **Facilitate Guided Practice and Application:** After modelling, we provide opportunities for pupils to apply their knowledge through **guided practice**, allowing them to solve problems using the base knowledge as a schema.
4. **Contextualise Learning:** We support pupils in connecting **theoretical concepts to real-world applications** to develop a practical understanding of how digital systems work and how to put this knowledge to use through programming.
5. **Utilise a Concept-Driven Curriculum:** We employ a curriculum centred on **key computing concepts** to build deep and transferable understanding, ensuring pupils can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems.

Curriculum Sequencing Rationale

In order to achieve an increasingly sophisticated understanding of computer science, topics have been intelligently sequenced based on the following rationale:

- **Foundation Building Through Direct Instruction**: We begin with foundational concepts such as understanding the principles of information and computation, how digital systems work, and basic programming skills. These are taught through direct instruction and modelling to ensure all pupils have a solid base upon which to build more complex ideas.

- **Progressive Complexity with Guided Practice**: Topics are arranged to gradually increase in complexity. Pupils apply their knowledge through **guided practice**, allowing them to make connections between concepts and develop problem-solving skills using their established schemas.
- **Interdisciplinary Links**: We incorporate elements of mathematics, science, and design and technology to provide a holistic understanding of how computer science interacts with other disciplines. Pupils learn to understand key algorithms, use logical reasoning, and apply mathematical concepts such as binary numbers and Boolean logic.
- **Real-World Applications**: We sequence topics to align with **current technological trends and societal needs**, making learning relevant and engaging. Pupils undertake projects that involve selecting, using, and combining multiple applications to achieve challenging goals, including collecting and analysing data and meeting the needs of known users.

Curriculum Building Blocks

To maximise the impact of our curriculum, the following elements have been selected as 'building blocks':

- **Direct Instruction and Modelling**: Clear teaching of concepts and procedures, providing pupils with the necessary knowledge and skills. Teachers model problem-solving processes, programming techniques, and computational methods.
- **Structured Practice and Application**: Pupils engage in structured activities that reinforce learning. **Guided practice** transitions to independent practice as pupils become more proficient.
- **Computational Thinking Skills**: Developing problem-solving strategies, logical reasoning, and algorithmic thinking. Pupils analyse problems in computational terms and gain practical experience in writing computer programs.
- **Understanding Digital Systems**: Knowledge of hardware and software components that make up computer systems, how they communicate, and how instructions are stored and executed.
- **Digital Literacy and Ethical Use**: Ensuring pupils become responsible, competent, confident, and creative users of information and communication technology. Pupils learn to use technology safely, respectfully, responsibly, and securely.

Addressing Social Disadvantage and Inclusivity

The computer science curriculum addresses social disadvantage by bridging gaps in pupils' knowledge and skills:

- **Equitable Access to Knowledge**: Through direct instruction, all pupils receive the same **high-quality teaching**, ensuring that everyone has access to the foundational knowledge required to succeed.
- **Supportive Learning Environment**: We provide additional support where needed, including resources and interventions for pupils who may require extra help in understanding key concepts.
- **Inclusivity Through Representation**: We highlight contributions from diverse cultures and communities within the field of computer science, promoting respect for diversity and encouraging all pupils to see themselves as potential contributors to the field.

Personal Development and Christian Values

We believe that computer science contributes to the personal development of pupils at St. George's:

- **Building Confidence Through Mastery**: By providing clear instruction and opportunities for success, pupils develop confidence in their abilities, reflecting the Christian value of hope.
- **Developing Resilience**: Pupils learn to persevere through challenging tasks, such as debugging code, which builds personal resilience.
- **Promoting Ethical Understanding**: Discussions on topics like data privacy and digital citizenship help pupils consider the moral implications of technology, fostering spiritual and moral development.
- **Encouraging Collaboration and Respect**: Group projects

Curriculum Sequencing

Sequencing		
Year 7	**Term 1**	**Introduction to the Computer System** **Computational Thinking and Algorithms with Block Programming**
	Term 2	**Binary Numbers** **Spreadsheet Modelling**
Year 8	**Term 1**	**Computer Systems Architecture** **Bitmap and Vector Images**
	Term 2	**Boolean Logic** **Block Programming Advanced**
Year 9	**Term 1**	**Cybersecurity** **Multimedia Production**
	Term 2	**Text Based Programming**
Year 10	**Term 1-2**	**OCR Paper 1 -** 1.2 Memory, 1.1 Systems Architecture, 1.5 System Software, 1.3 Networks **OCR Paper 2 -** 2.5 The IDE, 2.2 Programming Fundamentals,
	Term 3	**OCR Paper 1 -** 1.3 Network Protocols, 1.4 Network Security, 1.2 Data Storage **OCR Paper 2 -** 2.3 – Producing Robust Programs
Year 11	**Term 1**	**OCR Paper 1 -** 1.6 Ethical, legal, cultural and environmental impacts of digital technology **OCR Paper 2 -** 2.1 Algorithms, 2.4 Boolean logic, 2.5 Programming Languages
	Term 2-3	**OCR Paper 1 -** exam revision **OCR Paper 2 -** exam revision