St Philip Howard Catholic Voluntary Academy
Department Planning 2025 – 26

Long Term Mapping 2025 – 26

Computer Science – Yr 10

Subject Intent/ Aims:

At St Philip Howard the Computer Science department provides a high quality computing education that challenges the pupils to use an apply computational thinking and creativity to understand how they can have impact in the wider world through Computer Science.

The core aspects of the computer science curriculum are to support the pupils to develop an understanding of key computational principles; allowing them to learn how digital computer systems work and put this knowledge to use through the progressive use of programming.

The subject's intent is for pupils to build on the knowledge and skills each year as they progress from year 7 to year 11; with the overall aim being that the pupils will leave the school knowing and appreciating the opportunity they were given to learn and develop in an engaging subject that has a huge impact of the wider world.
As well as the Computer Science content delivered through the curriculum there is also an intention to ensure that pupils are given the chance to become digitally literate and be able to express themselves through the key aspects of information and communication technology.

The Computer Science department has a programme of study the follows the aims of the national curriculum. Within this, pupils are given the opportunity to learn how to understand and apply basic principles of computer science, analyse problems whilst confidently providing solutions, and acquire competency in using information and communication technology.

The overall intention of the computer science department at St Philip Howard to provide the pupils with a safe and engaging learning environment, that will foster a love for learning computer science and acquire a wide range of knowledge and skills that could have a huge benefit on their lives in and out of school.

St Philip Howard Catholic Voluntary Academy
Department Planning 2025 – 26

Key Concepts - Advent	Key Concepts - Lent	Key Concepts - Pentecost	
Term 1 and 2	Term 1 and 2	Term 1	Term 2
Interleaved Learning Computer Systems Revisit (Yr 9) Programming basics	**Interleaved Learning** Data Representation Programming	**Interleaved Learning** Logic Gates & Defensive Programming Design	**Exam Preperations**
National Curriculum Coverage	**National Curriculum Coverage**	**National Curriculum Coverage**	
✓ Apply analytical, problem-solving, design, and computational thinking skills. ✓ Gain comprehensive understanding of computer workings, including hardware and software.	✓ Apply analytical, problem-solving, design, and computational thinking skills. ✓ Gain comprehensive understanding of computer workings, including hardware and software.	✓ Apply analytical, problem-solving, design, and computational thinking skills. ✓ Gain comprehensive understanding of computer workings, including hardware and software.	❖ All previous
Components	**Components**	**Components**	
Computer Systems Revisit Explore CPU architecture, operating systems, memory management, and utility software to understand computer systems **Programming Basics** Programming fundamentals encompass variables, operators, control flow, data types, language levels, translators, compilers, interpreters, and IDE tools.	**Programming** The component covers additional programming techniques like string manipulation, random number generation, and working with Boolean operators in a high-level language within a classroom. **Data Representation** The component covers data representation, including converting between decimal, binary, and hexadecimal, adding binary integers with overflow, representing characters with binary codes, image representation with pixels in binary, sound sampling and storage in digital form, and types of compression (lossy and lossless).	**Logic Gates** The component requires students to understand logic gate truth tables and symbols, create and modify logic diagrams and truth tables for given scenarios, and work with multiple gates in a logic diagram. **Defensive Programming Design** The component involves learning defensive design and programming techniques, including anticipating misuse, authentication, input validation, maintainability, error identification, and iteration using loops.	Prepare for exams by revisiting and reviewing various computer science topics such as computer systems, programming basics, data representation, logic gates, and defensive programming design.

St Philip Howard Catholic Voluntary Academy
Department Planning 2025 – 26

HO Knowledge	HO Knowledge	HO Knowledge	
Computer Systems Revisit ❖ Critical analysis of CPU performance factors. ❖ Evaluation of memory management strategies. ❖ Problem-solving in optimizing system performance and troubleshooting. **Programming Basics** ❖ Applying logical thinking and analytical skills to identify and solve programming challenges. ❖ Evaluating code, identifying errors, and making informed decisions to optimize program performance. ❖ Thinking outside the box to develop innovative solutions and approaches in programming tasks.	**Programming** ❖ Applying string manipulation techniques, random number generation, and Boolean operators to solve programming challenges effectively. ❖ Analyzing and evaluating the appropriate use of string manipulation, random number generation, and Boolean operators to optimize program functionality. ❖ Utilizing string manipulation, random number generation, and Boolean operators innovatively to develop unique and efficient program solutions. **Data Representation** ❖ Analyzing data representations to convert between different formats effectively. ❖ Handling overflow errors when adding binary integers and optimizing file size and quality through compression techniques. ❖ Assessing the impact of various factors on image and sound quality, and making informed choices based on desired outcomes.	**Logic Gates** ❖ Applying logical thinking to analyze truth tables, recognize patterns, and make deductions based on given scenarios. ❖ Using problem-solving skills to create, modify, and complete logic diagrams and truth tables. ❖ Engaging in critical thinking to evaluate the effectiveness of logic diagrams and truth tables and propose improvements. **Defensive Programming Design** ❖ Using critical thinking to anticipate possible problems and protect a program from misuse. ❖ Finding smart solutions for handling invalid data, identifying errors, and creating secure authentication methods. ❖ Using careful analysis to improve the program's organization, readability, and efficiency through subprograms, clear naming, indentation, and helpful comments.	✓ Understanding computer systems, programming concepts, data representation, logic gates, and defensive programming to improve them. ✓ Using programming skills to solve problems and fix mistakes in programs. ✓ Evaluating computer systems, code, and data methods for speed, reliability, and safety. ✓ Combining knowledge from different areas to create innovative solutions. ✓ Learning effective study techniques, time management, and exam preparation in computer science.
Composite Skills	**Composite Skills**	**Composite Skills**	
Computer Systems Revisit 1. Analyzing CPU components and troubleshooting performance issues. 2. Understanding computer system architectures. 3. Understanding operating system functionalities for system operation.	**Programming** 1. Learning to concatenate and slice strings effectively. 2. Implementing the generation of random numbers within a given range. 3. Applying AND, OR, and NOT operators for logical evaluations and decision-making. 4. Practicing coding and implementing additional	**Logic Gates** 1. Understanding the functions and truth tables of different logic gates. 2. Recognizing and understanding the symbols used for each logic gate. 3. Building and modifying logic diagrams by selecting appropriate	1.Learn how computers work and what they do. 2. Understand the foundations of coding and how to write instructions for computers. 3. Discover interesting things

4. Understanding memory usage and understanding virtual memory. 5. Understanding utility software tools for system maintenance and optimization. **Programming Basics** 1. Using variables, operators, and control flow to write code. 2. Working with different types of data effectively. 3. Recognizing the difference between user-friendly and low-level programming languages. 4. Understanding how translators, compilers, and interpreters help run programs. 5. Making use of helpful tools and features in an Integrated Development Environment (IDE).	techniques in a high-level language. 5. Applying learned techniques in a classroom setting for practical programming exercises. **Data Representation** 1. Mastering the conversion between decimal, binary, and hexadecimal. 2. Handling addition of binary integers while considering overflow. 3. Understanding binary codes used to represent characters (e.g., ASCII or Unicode). 4. Grasping the representation of images using binary pixels and the impact of color depth and resolution on quality. 5. Learning about sampling, digital storage, and compression techniques for sound, including factors like sample rate, duration, and bit depth.	gates and making connections. 4. Interpreting and constructing truth tables to illustrate logical behaviour. 5. Working with multiple logic gates to represent complex logical relationships within a logic diagram. **Defensive Programming Design** 1. Thinking ahead to prevent program misuse and keeping it secure. 2. Checking and ensuring that only correct data is used. 3. Using clear names, organizing into smaller parts, and adding comments for easier understanding. 4. Identifying and resolving mistakes in the program's code. 5. Mastering the use of loops to repeat tasks based on specific conditions.	you can do with code to make programs more exciting. 4. Learn different ways to show and understand numbers, pictures, and sounds using computers. 5. Solve tricky puzzles that involve thinking logically and finding the right connections.

Final composition/ Deliberate Practice	Final composition/ Deliberate Practice	Final composition/ Deliberate Practice	
○ End of half term assessments to check specific topics in half term. ○ End of term assessments to check understanding off all topics in an entire term.	○ End of half term assessments to check specific topics in half term. ○ End of term assessments to check understanding of all topics since September	○ End of half term assessments to check specific topics in half term. ○ End of term assessments to check understanding of all topics since September	○ End of term assessments to check understanding of all topics since September
Assessment/s (Formative and Summative)	**Assessment/s (Formative and Summative)**	**Assessment/s (Formative and Summative)**	
✓ RRR ✓ Key terms tests ✓ Quizzes ✓ Flipped homework activities ✓ End of Topic Exam	✓ RRR ✓ Key terms tests ✓ Quizzes ✓ Flipped homework activities ✓ End of Topic Exam	✓ RRR ✓ Key terms tests ✓ Quizzes ✓ Flipped homework activities ✓ End of Topic Exam	

St Philip Howard Catholic Voluntary Academy
Department Planning 2025 – 26

Key Terms		Key Terms		Key Terms		
Computer Systems 1. CPU components 2. Performance troubleshooting 3. System architecture 4. Efficient design 5. Operating system functionalities 6. Memory optimization 7. Virtual memory 8. System operation 9. Memory management 10. Utility software 11. System maintenance 12. Optimization techniques	**Programming Basic** 1. Variables 2. Operators 3. Control flow 4. Code 5. Data types 6. User-friendly programming languages 7. Low-level programming languages 8. Translators 9. Compilers 10. Interpreters 11. Integrated Development Environment (IDE) 12. Tools and features	**Programming** 1. String manipulation 2. Concatenate 3. Slice 4. Random number generation 5. Range 6. Boolean operators 7. AND 8. OR 9. NOT 10. High-level language programming	**Data Representation** 1. Data representation 2. Decimal 3. Binary 4. Hexadecimal 5. Overflow 6. Character encoding 7. ASCII 8. Unicode 9. Binary pixels 10. Color depth 11. Resolution 12. Compression (lossy and lossless)	**Logic Gate** 1. Logic gate 2. Truth table 3. AND gate 4. OR gate 5. NOT gate 6. Symbol representation 7. Logic diagram 8. Input 9. Output 10. Connection 11. Logical operation 12. Circuit behavior	**Defensive Programming Design** 1. Defensive design 2. Misuse anticipation 3. Authentication 4. Input validation 5. Maintainability 6. Subprograms 7. Naming conventions 8. Indentation 9. Commenting 10. Syntax errors 11. Logic errors 12. Iteration	All previous

Literacy/ Numeracy/ Cross-Curricular Links	Literacy/ Numeracy/ Cross-Curricular Links	Literacy/ Numeracy/ Cross-Curricular Links	
Computer Systems Literacy: Enhance critical thinking through CPU analysis for improved literacy skills. Numeracy: Optimize numerical data manipulation with system architecture for stronger numeracy skills. Cross-curricular: Solve interdisciplinary problems using operating systems, memory management, and utility software. **Programming basics** Literacy: Writing code using variables, operators, and control flow enhances literacy skills through logical thinking, problem-solving, and effective expression.	**Programming** Literacy: Learning to concatenate and slice strings effectively enhances literacy skills by improving the ability to manipulate and organize textual data. Numeracy: Implementing the generation of random numbers within a given range develops numeracy skills by applying mathematical concepts to create randomization in program outcomes. Cross-curriculum: Applying AND, OR, and NOT operators for logical evaluations and decision-making promotes cross-curricular learning, integrating computational thinking and problem-solving across different subject areas.	**Logic Gates** Literacy: Understanding logic gate functions and truth tables enhances comprehension of complex logical concepts. Numeracy: Recognizing logic gate symbols and working with truth tables applies logical reasoning and mathematical thinking to binary data. Cross-curricular: Building logic diagrams, interpreting truth tables, and using multiple gates integrate literacy, numeracy, problem-solving, and	Literacy: Reading and understanding code, programming concepts, and technical documentation. Numeracy: Working with numbers, data representation, and mathematical operations in programming. Cross-curriculum: Applying computational thinking to problem-solving in various subjects, developing logical reasoning skills, and

Numeracy: Working with different data types develops numeracy skills by analyzing, interpreting, and manipulating data in various formats. Cross-curriculum: Recognizing programming language differences promotes cross-curricular learning, connecting technological literacy with critical thinking, computational literacy, and problem-solving skills.	**Data Representation** Literacy: Converting between decimal, binary, and hexadecimal improves understanding of different number systems. Numeracy: Adding binary integers while considering overflow develops mathematical skills within the binary system. Cross-curriculum: Understanding binary character codes, image representation, and sound sampling techniques integrate literacy and numeracy with technology and multimedia concepts.	technology skills. **Defensive Programming Design** Literacy: Planning ahead, organizing code, and fixing mistakes improve our problem-solving, communication, and critical thinking skills. Numeracy: Checking data, fixing errors, and using loops enhance our math skills, pattern recognition, and problem-solving abilities. Cross-curricular: Thinking critically, communicating clearly, and staying organized benefit not only computer science but also other subjects like math, language arts, and problem-solving tasks.	integrating computer science knowledge into different areas of study.

SMSC	BV	RSHE
✓ *There will be multiple opportunities for students develop spiritually; being creative in their learning with the different systems that they will create and programs, they will cultivate.* ✓ *The high expectations placed on the student from the school and department mean that pupils would regularly be made aware of the right and wrong morally.* ✓ *Pupils are expect to share the views morally on the different topics but also show respect and appreciate others in the classroom.* ✓ *The majority of topics will give the students opportunity to develop their social skills; some task will require students to collaborate with others.*	✓ *Students will further develop their knowledge of using the internet and social media.* ✓ *Students will be taught to fully appreciate other students viewpoints and the importance of being respectful when online as a digital citizen.* ✓ *Students will be taught the importance of selecting valid information from reliable sources for any presentation tasks that they do.* ✓ *Students are taught how to contribute to life in modern Britain by learning about the history of computing.* ✓ *Students will learning how to display British Values to use the internet and social media positively.*	✓ *The students will be taught about how to be safe online and the dangers.* ✓ *The students will be made aware of online relationships and the sexual issues that may arise.* ✓ *The students will be regularly conversed on their physical and mental health when overusing computers.*

St Philip Howard Catholic Voluntary Academy
Department Planning 2025 – 26

Adaptive Curriculum Content Programming	**Adaptive Curriculum Content** Further Data Representation and Logic	**Adaptive Curriculum Content** Networks	**Adaptive Curriculum Content** App Project	**Adaptive Curriculum Content** Exam

✓ Lesson job lists.
✓ Time taken to work on specific programming techniques is adapted accordingly.
✓ High achieving classes may be introduced to some topics from the following year, this is judged on class analysis.
✓ Adapted handouts for practical tasks.
  ○ Full versions
  ○ Partially complete
✓ Extended time provided for certain students.
✓ The end of topic online exam modified to reflect the topics covered by certain classes and ability levels.
	✓ Lesson job lists.
	✓ Adapted content of logic covered based on understanding.
	✓ Adapted handouts.
	✓ Not all parts of binary (math's) with be covered by all groups based on numeracy ability.
	✓ Calculators will be used for some students.
	✓ The end of topic online exam modified to reflect the topics covered by certain classes and ability levels.
		✓ Lesson job lists.
		✓ Adapted handouts.
		✓ Expectations of detail in work is varied based on ability.
			✓ Lesson job lists.
			✓ Expectations around the number of specific tasks in the project is adapted.
			✓ Expectations around number of explanations on tasks is based on ability levels.
			✓ Examples of projects completed for different ability levels.
				✓ Lesson job lists.
				✓ Adapted revision material
				  ○ Ability level specific.
				✓ Assessments adapted to cater for the students ability and what they have covered specifically in the year.

Adaptive Implementation Practices This is a summary of the practices used throughout the department/curriculum in line with school requests.			
Differentiated Instruction: *Tailoring class instructions to meet the diverse needs of students by providing varied materials, activities, and assessments.*	***Scaffolded Instruction:*** *Break down complex concepts into smaller, more manageable steps, providing additional support and guidance as students' progress through the material.*	***Formative Assessment:*** *Use ongoing assessments, such as quizzes, discussions, and peer reviews, to continuously monitor student progress and provide timely feedback.*	***Self-Paced Learning Job Lists:*** *Create self-paced lesson job lists or learning paths that allow students to progress through the lessons at their own speed, enabling them to take ownership of their learning process.*