Thomas Mills
High School & Sixth Form

# CURRICULUM OVERVIEW: COMPUTER SCIENCE

Our Computer Science curriculum at Thomas Mills High School & Sixth Form aims to provide pupils with a broad understanding and depth of knowledge in computing, building directly upon the foundational skills and concepts introduced in Key Stage 3. We are committed to fostering computational thinking, problem-solving abilities and a deep understanding of how technology works, ensuring that pupils are not only informed users of technology but also creative and responsible digital citizens.

Building upon the strong programming foundation from Year 9 (basic constructs, pseudocode, structured programs and string/number handling) and Year 10 (iteration and user input), Year 11 delves into advanced programming techniques, specifically focusing on data structures such as arrays and lists. Furthermore, leveraging the understanding of data representation from Year 10 and networks and cybersecurity from Year 9, Year 11 will consolidate knowledge of computer systems including Boolean logic, CPU architecture and various memory and storage types. We will also build upon the algorithmic thinking introduced in previous years by studying searching and sorting algorithms in detail.

Our goal is to prepare pupils for their next steps, whether that be further academic study in Computer Science or related fields, apprenticeships, or future careers in the ever-evolving digital landscape. We equip them with the essential knowledge and practical skills to navigate and contribute positively to the digital world, empowering them to be confident, capable and ethical participants in a technology-driven society.

**Qualification:** AQA GCSE Computer Science 8525

Thomas Mills
High School & Sixth Form



## Year 11

| Term | Topic | Knowledge and Skills | Useful Links |
|---|---|---|---|
| 1 | **Unit 2H:** Learn how to use arrays and lists | This unit introduces pupils to data structures, specifically arrays and lists, which are ways to organise and store collections of data in a program.<br><br>**Knowledge:**<br>• Definition of a data structure.<br>• Definition of a list and an array.<br>• Differences between lists and arrays.<br>• Understanding of 2D arrays and lists.<br><br>**Skills:**<br>• Using a list in a program.<br>• Appending to a list.<br>• Traversing a list of elements.<br>• Writing algorithms in Python involving arrays and lists. | **How you can help at home:**<br>• Encourage your child to try out simple array/list-based Python programs using online interpreters such as Edublocks.<br>• Discuss how video games use lists/arrays for things like character inventories or high scores.<br>• If your child has a collection (e.g. - trading cards, books), talk about how they could organise it like a list or a 2D array.<br><br>**AQA Specification:**<br>• 3.2.6<br>• AQA \| Computer Science \| GCSE \| GCSE Computer Science<br><br>**Links:**<br>• BBC Bitesize: Programming languages - AQA<br>• Ada Computer Science: Data structures<br>• W3Schools Python Arrays Tutorial<br>• W3Schools Python Lists Tutorial<br>• Craig n Dave: SLR09 - Advanced programming concepts<br>• PG Online Textbook: Chapter 2: 2A.4<br>• Seneca Learning: Computer Science |

Thomas Mills
High School & Sixth Form



| | |
|---|---|
| **Assessment** | Your child will have a mid-point multiple-choice quiz and a summative assessment (made up of past paper examination questions) to evaluate their understanding of user input (Unit 2G) and data structures (arrays and lists) (Unit 2H). Past paper questions will be a mix of 'shade the lozenge', short answer and longer answer coding-based questions. |

Thomas Mills
High School & Sixth Form

# CURRICULUM OVERVIEW: COMPUTER SCIENCE

| | | | |
|---|---|---|---|
| **1** | **Unit 1A:** Algorithms | This unit focuses on algorithms, which are detailed, step-by-step instructions for solving problems. Your child will explore different search algorithms, such as linear and binary search, to efficiently find specific items within data sets. The unit also covers sorting algorithms like Bubble Sort and Merge Sort, which are used to arrange data in a particular order. A key skill developed here is "tracing" algorithms, which involves following the steps of an algorithm to understand its execution and identify any issues.<br><br>**Knowledge:**<br>• Difference between algorithms and computer programs.<br>• Algorithms defined as written descriptions, flowcharts and code.<br>• Flowchart symbols (start, end, input, output, subroutine).<br>• Why computers often need to search and sort data.<br>• How linear search and binary search is used.<br>• How Bubble Sort and Merge Sort works.<br>• Factors that could influence the efficiency of a bubble sort implementation.<br><br>**Skills:**<br>• Analysing and creating flowcharts.<br>• Translating a flowchart into a program sequence.<br>• Designing a flowchart for a program.<br>• Performing a linear search and binary search. | **How you can help at home:**<br>• Identify daily routines (recipes, sorting laundry) as algorithms and discuss their steps.<br>• Practice drawing simple flowcharts for basic activities.<br>• Use physical objects (cards) to demonstrate search and sort algorithms.<br>• Ask your child how they would quickly find a word in a dictionary to illustrate efficiency.<br><br>**AQA Specification:**<br>• 3.1.1 to 3.1.4<br>• AQA \| Computer Science \| GCSE \| GCSE Computer Science<br><br>**Links:**<br>• BBC Bitesize: Fundamentals of algorithms - AQA<br>• BBC Bitesize: Searching and sorting algorithms - AQA<br>• Ada Computer Science: Searching algorithms<br>• Ada Computer Science: Sorting algorithms<br>• Craig n Dave: SLR07 - Algorithms<br>• PG Online Textbook: Chapter 1: 1.4 to 1.5<br>• Seneca Learning: Computer Science |

| | | | |
|---|---|---|---|
| | | • Comparing the features of linear and binary search and deciding suitability.<br>• Using a trace table to walk through code and to detect and correct errors.<br>• Interpreting and tracing the code for linear search and binary search.<br>• Interpreting and tracing the code for a Bubble Sort.<br>• Performing a Merge Sort. | |
| **Assessments** | | Your child will have a mid-point multiple-choice quiz and a summative assessment (made up of past paper examination questions) to evaluate their understanding of algorithms. Additionally, your child will have a summative assessment assessing all content taught to date, focusing on specification points 3.1 to 3.8 (excluding 3.4). Past paper questions will be a mix of 'shade the lozenge', short answer and longer answer coding/written questions. This should be read in conjunction with the AQA subject content within the specification. | |

Thomas Mills
High School & Sixth Form



| | | | |
|---|---|---|---|
| **1** | **Unit 2I:** Robust and Secure Programming | This unit focuses on developing good programming practices and how to write structured code. Your child will understand the concept of "scope" in programming, which dictates where variables can be accessed. A significant part of the unit is dedicated to ensuring programs are reliable and robust.<br><br>**Knowledge:**<br>• Subroutine concepts: purpose, parameters, decomposition, advantages and the distinction between functions and procedures.<br>• Local vs. global variables and their appropriate use.<br>• Constants and the structured programming approach, including its benefits.<br>• Program testing: iterative testing, types of test data (normal, boundary, erroneous) and identifying syntax and logic errors.<br>• The need for validation checks and common techniques.<br>• Authentication routine principles (username/password).<br><br>**Skills:**<br>• Implementing and utilising subroutines (procedures and functions) with parameters and return values.<br>• Applying structured programming principles to design and create programs.<br>• Identifying and correcting syntax and logic errors. | **How you can help at home:**<br>• Encourage your child to try out simple Python programs using online interpreters such as Edublocks.<br>• Help your child divide complex tasks into smaller steps, mirroring subroutines.<br>• Discuss how apps handle incorrect user input to highlight validation.<br>• Talk about strong passwords and why online authentication is crucial.<br>• Have them explain their code aloud; it often helps them find errors.<br><br>**AQA Specification:**<br>• 3.2.10 to 3.2.11<br>• AQA \| Computer Science \| GCSE \| GCSE Computer Science<br><br>**Links:**<br>• BBC Bitesize: Fundamentals of algorithms - AQA<br>• BBC Bitesize: Further programming language operations - AQA<br>• Ada Computer Science: Subroutines<br>• Ada Computer Science: Testing<br>• Craig n Dave: SLR09 - Advanced programming concepts<br>• Craig n Dave: SLR10 - Robust and secure programming<br>• PG Online Textbook: Chapter 2: 2B.1 to 2B.4 |

| | | | |
|---|---|---|---|
| | | • Selecting appropriate test data and performing iterative testing.<br>• Implementing data validation and basic authentication routines. | • Seneca Learning: Computer Science |
| **Assessment** | | Your child will have a mid-point multiple-choice quiz and a summative assessment (made up of past paper examination questions) to evaluate their understanding of secure and robust programming. Past paper questions will be a mix of 'shade the lozenge', short answer and longer answer coding-based questions. | |

| | | | |
|---|---|---|---|
| **2** | **Unit 4:** Computer Systems | This unit explores the foundations of computing, starting with Boolean logic, logic gates and truth tables. Your child will learn about the Central Processing Unit (CPU), its components, the fetch-decode-execute cycle and factors influencing its performance. The unit also covers programming languages (high and low-level) and their translators (compilers, interpreters, assemblers). Finally, your child will distinguish between hardware and software, examining embedded systems, main memory (RAM/ROM) and various secondary storage options.<br><br>**Knowledge:**<br>• Logic gates (AND, NOT, OR, XOR), their symbols, truth tables. Boolean operations, notation and expressions.<br>• Basic CPU components, their roles in computation and the fetch-decode-execute cycle.<br>• Factors influencing CPU performance (clock speed, cache, cores).<br>• Relationship between assembly and machine code.<br>• Differences between high- and low-level languages and the need for and types of translators (compilers, interpreters, assemblers).<br>• Definition and roles of embedded systems, system software (OS, utilities) and the distinction between hardware and software.<br>• Characteristics and roles of RAM, ROM, main memory and cache. Need for and types of | **How you can help at home:**<br>• Briefly explain components of devices they use (CPU, hard drive).<br>• Use light switches to explain basic Boolean logic (AND, OR).<br>• Discuss specifications (CPU, RAM, etc) when considering new electronics.<br>• Explain compilers vs. interpreters using a "book translator" example (e.g. - with a compiler the book is translated once into a different language and re-written vs. an interpreter translating word-by-word each time).<br>• Discuss cloud storage benefits and risks for their online data.<br><br>**AQA Specification:**<br>• 3.4.1 to 3.4.5<br>• AQA \| Computer Science \| GCSE \| GCSE Computer Science<br><br>**Links:**<br>• BBC Bitesize: Computer systems - AQA<br>• BBC Bitesize: Classification of programming languages and translators - AQA<br>• BBC Bitesize: Systems architecture - AQA<br>• Ada Computer Science: Hardware<br>• Ada Computer Science: Software<br>• Ada Computer Science: Systems architecture<br>• Ada Computer Science: Boolean logic |

| | | | |
|---|---|---|---|
| | | secondary storage (solid-state, optical, magnetic), how they store/read data and the need for cloud storage.<br><br>**Skills:**<br>• Designing and constructing logic circuits with truth tables and Boolean expressions, including XOR. Evaluating Boolean expressions and applying logic to problem-solving.<br>• Discussing factors affecting CPU performance.<br>• Comparing high/low-level languages and evaluating compilers, interpreters and assemblers.<br>• Comparing embedded vs. general-purpose systems. Explaining roles of OS, utilities and main memory.<br>• Applying knowledge of storage devices to compare and select appropriate types.<br>• Explaining cloud storage.<br>• Selecting components and evaluating a computer's suitability for specific tasks. | • Ada Computer Science: Memory and storage<br>• Ada Computer Science: Operating systems<br>• Ada Computer Science: Programming languages<br>• Ada Computer Science: Translators<br>• Craig n Dave: SLR01 - Systems architecture<br>• Craig n Dave: SLR02 - Memory and storage<br>• Craig n Dave: SLR05 - Hardware and software<br>• Craig n Dave: SLR11 - Boolean logic<br>• Craig n Dave: SLR12 - Classification of programming languages<br>• PG Online Textbook: Chapter 4: 4.1 to 4.7<br>• Seneca Learning: Computer Science |
| **Assessments** | | Your child will have two mid-point multiple-choice quizzes and a summative assessment (made up of past paper examination questions) to evaluate their understanding of computer systems. Additionally, your child will have two summative assessments assessing all content taught to date. Paper 1 (the programming paper) focuses on specification points 3.1 to 3.2. Paper 2 (the theoretical paper) focuses on specification points 3.3 to 3.8. Past paper questions will be a mix of 'shade the lozenge', short answer and longer answer coding/written questions. This should be read in conjunction with the AQA subject content within the specification. | |