

Computer Science Curriculum Intent

Computer Science GCSE

At GCSE, pupils explore how computer systems function, from physical hardware to the software and programs users interact with. They learn to design, build and adapt computer systems, applying computational thinking and programming creatively to solve real-world problems. The course develops logical thinking, problem-solving and data-handling skills, while ensuring pupils understand the risks of storing personal information electronically and the strategies used to maintain security.

Computer Science A Level

At A Level, students deepen their understanding of how computers operate and develop advanced computational and problem-solving skills. Through the non-exam assessment, they design and implement a bespoke software solution using techniques such as object-oriented programming, event-driven programming and graphical user interfaces. The course prepares students for degree study, apprenticeships and careers in software development and related computing fields.

Year	Knowledge (Topics/Unit of work? What pupils will 'know'	Prior Knowledge (KS2/Previous learning)	Skills acquired What pupils will be able to 'do'	Concepts developed (What pupils will understand?)	Key Vocabulary (Tier 3)	Writing & Communication
7 Term 1						
7 Term 2						
7 Term 3						
8 Term 1						
8 Term 2						
8 Term 3						
9 Term 1						
9 Term 2						
9 Term 3						

Year	Knowledge (Topics/Unit of work? What pupils will 'know'	Prior Knowledge (KS2/Previous learning)	Skills acquired What pupils will be able to 'do'	Concepts developed (What pupils will understand?)	Key Vocabulary (Tier 3)	Writing & Communication
10 Term 1	Data representation and programming - Binary, decimal and hexadecimal number systems and conversions. - Binary addition, subtraction and shifts. - ASCII and Unicode character encoding. - Bitmap images: pixels, resolution and colour depth. - Sound: sampling, sample rate and bit depth. - Units of data and lossless/lossy compression. - Ongoing Python: sequence, selection, iteration, data types, variables and input/output.	- KS3 binary and basic data units. - Prior Python experience using variables, input and selection.	- Convert between number bases and perform binary arithmetic. - Calculate image and sound file sizes. - Choose suitable compression methods. - Write, test and debug short Python programs.	- Why computers use binary. - Trade-offs between quality, file size and compression. - How data is represented as binary patterns. - Readable, maintainable and testable code.	- binary - denary - hexadecimal - bit - byte - pixel - resolution - colour depth - sample rate - bit depth - compression	Do now tasks: Binary/hex conversion, file-size quick questions, or spot-the-error Python starter. Classwork tasks- written: Complete calculation questions, compression comparisons and short exam-style explanations. Classwork tasks- independent research: Research where images, sound and compression are used in real systems. Homework tasks: Past-paper data representation questions plus a short Python practice task. Guided reading task: Read a worked example on image/sound representation and annotate key formulae.

Year	Knowledge (Topics/Unit of work? What pupils will 'know'	Prior Knowledge (KS2/Previous learning)	Skills acquired What pupils will be able to 'do'	Concepts developed (What pupils will understand?)	Key Vocabulary (Tier 3)	Writing & Communication
10 Term 2	Algorithms and computer systems • Decomposition, abstraction and algorithm design. • Linear and binary search; bubble, insertion and merge sort. • Pseudocode and flowcharts. • CPU components, registers, cache and the fetch-decode-execute cycle. • Factors affecting performance: clock speed, cores and cache. • Embedded systems; RAM, ROM and secondary storage. • Programming languages and translators: compiler, interpreter and assembler.	• Sequence, selection and iteration. • Binary representation and basic computer hardware.	• Trace, compare and refine algorithms. • Represent solutions using pseudocode and flowcharts. • Describe CPU components and the fetch-decode-execute cycle. • Evaluate storage and embedded systems for scenarios.	• Why decomposition and abstraction reduce complexity. • Different algorithms can solve the same problem with different efficiency. • How hardware and software interact. • Storage choices involve capacity, speed, durability, portability, reliability and cost.	• algorithm • decomposition • abstraction • linear search • binary search • bubble sort • merge sort • CPU • ALU • register • cache • embedded system	Do now tasks: Trace-table starter, pseudocode gap-fill, CPU component match-up or sort/search recap. Classwork tasks- written: Write and refine algorithms; explain CPU, FDE cycle and storage choices in scenarios. Classwork tasks- independent research: Compare embedded systems or storage technologies used in everyday devices. Homework tasks: Complete algorithm traces and past-paper systems questions. Guided reading task: Read a CPU/FDE explanation and highlight inputs, processes and outputs.

Year	Knowledge (Topics/Unit of work? What pupils will 'know'	Prior Knowledge (KS2/Previous learning)	Skills acquired What pupils will be able to 'do'	Concepts developed (What pupils will understand?)	Key Vocabulary (Tier 3)	Writing & Communication
10 Term 3	Networks and cyber security • LANs and WANs; network performance factors. • Routers, switches, wireless access points, NICs and transmission media. • Client-server and peer-to-peer networks. • Internet, DNS, hosting, cloud and virtual networks. • Threats including malware, phishing, brute force, denial of service and data interception. • Social engineering and layered protection: firewalls, anti-malware, access levels, passwords, encryption, penetration testing and physical security.	• Computer systems and data transmission concepts. • Safe and responsible use of digital systems.	• Compare network models and explain DNS. • Select suitable network types for scenarios. • Identify threats and recommend proportionate prevention methods. • Evaluate advantages and disadvantages of cloud computing.	• Bandwidth and latency affect performance. • Network design depends on purpose, scale and security. • Social engineering exploits human behaviour. • Effective security uses multiple layers.	• LAN • WAN • bandwidth • latency • router • switch • WAP • NIC • DNS • cloud • malware • phishing • encryption • firewall	Do now tasks: Network hardware recall, threat/defence match-up, DNS ordering or key-term quiz. Classwork tasks- written: Answer scenario questions on networks, cloud computing and cyber-security controls. Classwork tasks- independent research: Investigate a recent cyber incident and identify threat, impact and prevention. Homework tasks: Create a revision summary for networks or complete cyber-security exam questions. Guided reading task: Read a case study on social engineering and annotate the layered defences used.

Year	Knowledge (Topics/Unit of work? What pupils will 'know'	Prior Knowledge (KS2/Previous learning)	Skills acquired What pupils will be able to 'do'	Concepts developed (What pupils will understand?)	Key Vocabulary (Tier 3)	Writing & Communication
11 Term 1	Relational databases, SQL and programming - Tables, records, fields, primary and foreign keys and relationships. - Database data types; validation and verification. - SQL: SELECT, FROM, WHERE, ORDER BY, AND/OR/NOT, LIKE, BETWEEN and IN. - Aggregate functions: COUNT, SUM, AVG, MIN and MAX; basic GROUP BY. - INNER JOIN across two related tables. - Ongoing programming: arrays, strings, subroutines, parameters, scope, validation, authentication, test plans and trace tables.	- Variables, data types, records and files. - Boolean logic and structured programming.	- Design simple relational tables and ER diagrams. - Write SQL queries to filter, sort, aggregate and join data. - Use one- and two-dimensional arrays. - Write procedures/functions and systematic test plans.	- Keys enforce relationships and support data integrity. - Relational design reduces duplication. - Validation and verification serve different purposes. - Modular programming improves maintainability.	- database - table - record - field - primary key - foreign key - relationship - validation - verification - SQL - query - aggregate - join	Do now tasks: SQL keyword recall, key-field match-up, validation/verification sorter or code trace. Classwork tasks- written: Design tables/ERDs and write SQL queries, JOINS, aggregates and test plans. Classwork tasks- independent research: Research how organisations use relational databases to reduce duplication. Homework tasks: SQL worksheet with increasing challenge and a short programming/debugging task. Guided reading task: Read an SQL worked example and annotate clauses, operators and expected output.

Year	Knowledge (Topics/Unit of work? What pupils will 'know'	Prior Knowledge (KS2/Previous learning)	Skills acquired What pupils will be able to 'do'	Concepts developed (What pupils will understand?)	Key Vocabulary (Tier 3)	Writing & Communication
11 Term 2	Algorithms, impacts of digital technology and exam preparation - Revision and comparison of searching and sorting algorithms. - Data Protection Act, Computer Misuse Act, Copyright, Designs and Patents Act, Creative Commons and Freedom of Information Act. - Ethical, cultural, social and environmental impacts of technology. - Energy use, electronic waste and sustainability. - Consolidation of Paper 1 programming and Paper 2 computing concepts.	- Year 10 algorithms, networks, cybersecurity and data representation. - Year 11 databases and programming.	- Compare algorithms and explain how they operate. - Apply legislation to realistic scenarios. - Discuss ethical, cultural and environmental impacts using evidence. - Answer timed exam questions and improve responses from feedback.	- Law, ethics and culture overlap but are not identical. - Digital technology creates both benefits and unintended consequences. - Algorithm choice depends on data and context.	- legislation - privacy - copyright - licence - ethical - cultural - environmental - sustainability - e-waste - linear search - binary search - merge sort	Do now tasks: Legislation quick-fire, sort/search comparison, command-word practice or misconception check. Classwork tasks- written: Write balanced 6-8 mark responses on legal, ethical and environmental impacts. Classwork tasks- independent research: Research one technology impact, e.g. AI, e-waste, surveillance or remote working. Homework tasks: Complete mixed past-paper questions and improve one response using the mark scheme. Guided reading task: Read a digital technology case study and identify legal, ethical and environmental points.
11 Term 3	Revision and reinforcement - Targeted revision of AQA GCSE Computer Science Paper 1 and Paper 2. - Programming practice, algorithm tracing and correction of misconceptions. - Retrieval practice across data representation, systems, networks, cybersecurity, databases and impacts.	All Year 10 and Year 11 course content.	- Complete timed papers and programming tasks. - Use mark schemes to diagnose gaps and improve answers. - Select and apply knowledge across synoptic questions.	- Exam questions often combine several topics. - Accurate terminology, clear working and contextual application secure marks.	- computational thinking - pseudocode - trace table - debugging - evaluation - application - justify	Do now tasks: Interleaved retrieval quiz, calculation drill, SQL starter or trace-table recap. Classwork tasks- written: Complete timed exam questions, redraft weak answers and build personal revision notes. Classwork tasks- independent research: Use specification checklists to identify independent revision priorities and resources. Homework tasks: Targeted past-paper practice based on mock/question-level analysis. Guided reading task: Annotate examiner reports or mark-scheme extracts to identify how marks are awarded.

Year	Knowledge (Topics/Unit of work? What pupils will 'know'	Assessment (KPIs: Diagnostic + Summative)	Cumulative Links (What is revisited?)	Adaptive Teaching (How all pupils access learning)	Future Links (KS4/KS5)
7 Term 1					
7 Term 2					
7 Term 3					
8 Term 1					
8 Term 2					
8 Term 3					
9 Term 1					
9 Term 2					
9 Term 3					
10 Term 1	Data representation and programming	- Diagnostic retrieval quiz from KS3. - Low-stakes quizzes and mini-whiteboard checks. - End-of-topic data representation test. - Programming tasks with testing/debugging evidence.	- KS3 binary, data units and introductory programming. - Programming constructs revisited throughout Years 10–11.	- Worked examples and conversion grids. - Scaffolded file-size formulae and partially completed code. - Vocabulary banks, paired tracing and extension problems.	Do now tasks: Binary/hex conversion, file-size quick questions, or spot-the-error Python starter. Classwork tasks- written: Complete calculation questions, compression comparisons and short exam-style explanations. Classwork tasks- independent research: Research where images, sound and compression are used in real systems. Homework tasks: Past-paper data representation questions plus a short Python practice task. Guided reading task: Read a worked example on image/sound representation and annotate key formulae.

Year	Knowledge (Topics/Unit of work? What pupils will 'know'	Assessment (KPIs: Diagnostic + Summative)	Cumulative Links (What is revisited?)	Adaptive Teaching (How all pupils access learning)	Future Links (KS4/KS5)
10 Term 2	Algorithms and computer systems	- Algorithm tracing and pseudocode checkpoints. - End-of-topic algorithms and systems assessments. - Structured homework using past-paper questions. - Formal Year 10 mock papers under timed conditions.	- Programming constructs and binary from Term 1. - Algorithm tracing is revisited in Year 11 and A Level.	- Chunked flowcharts and trace tables. - Physical/visual CPU models and labelled diagrams. - Model answers, guided comparisons and stretch efficiency questions.	Do now tasks: Trace-table starter, pseudocode gap-fill, CPU component match-up or sort/search recap. Classwork tasks- written: Write and refine algorithms; explain CPU, FDE cycle and storage choices in scenarios. Classwork tasks- independent research: Compare embedded systems or storage technologies used in everyday devices. Homework tasks: Complete algorithm traces and past-paper systems questions. Guided reading task: Read a CPU/FDE explanation and highlight inputs, processes and outputs.
10 Term 3	Networks and cyber security	- Network and cybersecurity scenario questions. - End-of-topic tests. - Year 10 mock reflection: WWW/EBI and action plan. - Assessment data used to plan intervention.	- Computer hardware and data transmission. - Boolean thinking used when evaluating threats and controls.	- Network diagrams and scenario cards. - Threat/defence matching activities. - Sentence stems for evaluation and accessible case studies.	Do now tasks: Network hardware recall, threat/defence match-up, DNS ordering or key-term quiz. Classwork tasks- written: Answer scenario questions on networks, cloud computing and cyber-security controls. Classwork tasks- independent research: Investigate a recent cyber incident and identify threat, impact and prevention. Homework tasks: Create a revision summary for networks or complete cyber-security exam questions. Guided reading task: Read a case study on social engineering and annotate the layered defences used.

Year	Knowledge (Topics/Unit of work? What pupils will 'know'	Assessment (KPIs: Diagnostic + Summative)	Cumulative Links (What is revisited?)	Adaptive Teaching (How all pupils access learning)	Future Links (KS4/KS5)
11 Term 1	Relational databases, SQL and programming	- SQL query tasks and database design assessment. - Programming assessment using arrays, subroutines and test plans. - Regular retrieval quizzes. - Year 11 mock Paper 1 and Paper 2 under timed conditions.	- Records, files, validation and Boolean expressions. - Programming knowledge from Year 10 is deepened.	- Pre-built tables and colour-coded keys. - SQL writing frames and query-building steps. - Code templates, live modelling and graduated challenge.	Do now tasks: SQL keyword recall, key-field match-up, validation/verification sorter or code trace. Classwork tasks- written: Design tables/ERDs and write SQL queries, JOINS, aggregates and test plans. Classwork tasks- independent research: Research how organisations use relational databases to reduce duplication. Homework tasks: SQL worksheet with increasing challenge and a short programming/debugging task. Guided reading task: Read an SQL worked example and annotate clauses, operators and expected output.
11 Term 2	Algorithms and impacts of digital technology	- Past-paper algorithm and legislation questions. - Extended-response assessment on impacts. - Timed mixed-topic papers. - Intervention informed by question-level analysis.	- Year 10 algorithms and cybersecurity. - Legislation linked to privacy, data and security.	- Algorithm animations and worked traces. - Law-to-scenario matching grids. - Writing frames for 6–8 mark responses and model answer annotation.	Do now tasks: Legislation quick-fire, sort/search comparison, command-word practice or misconception check. Classwork tasks- written: Write balanced 6-8 mark responses on legal, ethical and environmental impacts. Classwork tasks- independent research: Research one technology impact, e.g. AI, e-waste, surveillance or remote working. Homework tasks: Complete mixed past-paper questions and improve one response using the mark scheme. Guided reading task: Read a digital technology case study and identify legal, ethical and environmental points.

Year	Knowledge (Topics/Unit of work? What pupils will 'know'	Assessment (KPIs: Diagnostic + Summative)	Cumulative Links (What is revisited?)	Adaptive Teaching (How all pupils access learning)	Future Links (KS4/KS5)
11 Term 3	Revision and reinforcement	• Full or adapted examination papers. • Targeted diagnostic quizzes. • Teacher/pupil analysis of recurring errors. • Final examination readiness checks.	• All GCSE content is interleaved and revisited. • Priority topics selected from assessment evidence.	• Personalised revision lists. • Tiered practice by confidence and target grade. • Reduced scaffolding over time; alternative visual and written explanations.	Do now tasks: Interleaved retrieval quiz, calculation drill, SQL starter or trace-table recap. Classwork tasks- written: Complete timed exam questions, redraft weak answers and build personal revision notes. Classwork tasks- independent research: Use specification checklists to identify independent revision priorities and resources. Homework tasks: Targeted past-paper practice based on mock/question-level analysis. Guided reading task: Annotate examiner reports or mark-scheme extracts to identify how marks are awarded.

Year	Unit/Big idea	Disciplinary Thinking & Methods	Cumulative Links (What is revisited?)	Assessments (KPIs: Diagnostic + Summative)	Adaptive Teaching (How all pupils access learning)	Future Links (KS5)
12 Term 1	Programming, data representation and computer systems	• Use appropriate data types, control structures, subroutines, scope and exception handling. • Develop procedural and object-oriented programs; read/write files. • Convert number bases, use two's complement and calculate image/sound sizes. • Apply parity and checksums. • Classify software, compare translators, simplify Boolean expressions and explain operating-system functions.	• Builds directly on GCSE programming, data representation, logic and systems. • Programming is practised throughout Year 12.	• Homework and problem-solving challenges. • Low-stakes quizzes and mini-whiteboard checks. • End-of-topic past-paper assessments. • Shortened Paper 1 and Paper 2 mock with reflection/action plan.	• Code templates, live modelling and staged challenges. • Visual representations for binary, logic and OS processes. • Vocabulary banks, exemplars and AO-marked writing frames.	Do now tasks: Data type recall, binary/two complement questions, Boolean simplification or code prediction. Classwork tasks- written: Explain representation, translators, OS functions and programming constructs using exam language. Classwork tasks- independent research: Research real uses of compilers/interpreters, operating systems or image/sound encoding. Homework tasks: Isaac/Smart Revise retrieval plus programming and representation questions. Guided reading task: Read specification notes on programming/data representation and annotate required precision.
12 Term 2	Architecture, networking and theory of computation	• Explain registers, buses, stored-program concept and instruction execution. • Apply immediate and direct addressing modes. • Compare transmission methods and trace TCP/IP communication. • Calculate subnetting/IP information and propose security mitigations. • Construct finite-state machines, transition tables and regular expressions; simulate simple Turing machines.	• Extends GCSE CPU, networks, cybersecurity and algorithms. • Uses Boolean logic and binary from Term 1.	• Architecture and networking topic tests. • FSM/Turing machine problem sets. • Timed examination questions. • Retrieval quizzes using Smart Revise and Isaac Computer Science.	• Register/bus diagrams and worked assembly traces. • Layered network models and subnetting scaffolds. • State-machine templates and guided simulations. • Extension proofs and unfamiliar-context questions.	Do now tasks: Register recall, bus/function match-up, subnetting starter or FSM transition check. Classwork tasks- written: Trace FDE, networking layers, protocols, FSMs and Turing machine examples. Classwork tasks- independent research: Investigate network security or processor architecture in a chosen real-world system. Homework tasks: Architecture, networking and theory of computation past-paper questions. Guided reading task: Read an FSM or TCP/IP example and annotate each state/layer and transition/process.

Year	Unit/Big idea	Disciplinary Thinking & Methods	Cumulative Links (What is revisited?)	Assessments (KPIs: Diagnostic + Summative)	Adaptive Teaching (How all pupils access learning)	Future Links (KS5)
12 Term 3	Data structures, algorithms and Big Data	- Implement arrays, records, stacks, queues, trees, graphs, hash tables and dictionaries. - Choose suitable ADTs and trace push/pop, enqueue/dequeue and traversal. - Trace linear/binary search, merge sort and Dijkstra's algorithm. - Use Big-O to compare efficiency; discuss divide-and-conquer, optimisation and heuristics. - Evaluate Big Data quality, security, privacy and distributed/NoSQL approaches.	- Revisits GCSE algorithms and programming. - Applies abstraction, recursion and object-oriented design.	- Programming challenges and algorithm traces. - End-of-topic data structures/algorithms tests. - Big Data evaluation task. - Year 12 mock papers and post-mock action plan.	- Concrete visualisations of stacks, queues, trees and graphs. - Partially completed algorithms and trace tables. - Graduated programming tasks and model evaluations.	Do now tasks: Stack/queue operation starter, Big-O recall, graph/tree terminology or algorithm trace. Classwork tasks- written: Trace data-structure operations, compare algorithms and evaluate Big Data scenarios. Classwork tasks- independent research: Research Big Data use in health, retail, education or transport, including risks. Homework tasks: Implement or trace an ADT and complete algorithm-efficiency questions. Guided reading task: Read a worked Dijkstra/Big-O example and annotate decision points and complexity.
13 Term 1	Databases, functional programming, consequences of computing and NEA	- Normalise relational databases to 1NF, 2NF and 3NF and write SQL. - Use immutability, recursion, higher-order and first-class functions. - Analyse ethical, legal, cultural and social consequences of computing. - Complete NEA analysis, design, implementation, testing and evaluation using decomposition, abstraction and prototyping.	- Extends GCSE/Year 11 SQL and Year 12 programming. - Uses algorithms, data structures and software-development methods from Year 12.	- Formative NEA feedback at permitted milestones. - Database, functional programming and consequences tests. - Regular retrieval and examination questions. - Evidence against NEA assessment objectives.	- Normalisation worked examples and SQL scaffolds. - Side-by-side imperative/functional examples. - NEA milestone checklists and conferencing within assessment rules. - Writing frames for ethical/legal analysis.	Do now tasks: Normalisation starter, SQL recall, functional programming prediction or NEA milestone check. Classwork tasks- written: Normalise data, write SQL, compare paradigms and draft concise NEA evidence. Classwork tasks- independent research: Research ethical/legal consequences linked to the student NEA context. Homework tasks: NEA development tasks plus targeted database/functional programming practice. Guided reading task: Read exemplar NEA or ethical scenario extracts and annotate assessment requirements.

Year	Unit/Big idea	Disciplinary Thinking & Methods	Cumulative Links (What is revisited?)	Assessments (KPIs: Diagnostic + Summative)	Adaptive Teaching (How all pupils access learning)	Future Links (KS5)
13 Term 2	NEA completion, Paper 1 skeleton code and examination preparation	- Complete and evaluate the programmed solution and concise documentation. - Read, trace and explain the AQA skeleton program. - Amend and extend code under examination conditions. - Practise computational thinking, debugging and unseen programming.	- Integrates programming, algorithms and data structures from the whole course. - Revisits weak areas identified through assessment.	- Final NEA assessment. - Timed skeleton-code tasks. - Paper 1 and Paper 2 mock examinations. - Question-level analysis and targeted intervention.	- Chunked code walkthroughs and annotated program maps. - Graduated amendment tasks. - Targeted revision pathways and exam-writing scaffolds.	Do now tasks: Skeleton-code retrieval, code trace, error-spotting or key-subroutine explanation. Classwork tasks- written: Annotate skeleton code, explain program constructs and plan amendments/extensions. Classwork tasks- independent research: Research likely techniques used in similar programs and revise weak specification areas. Homework tasks: Timed skeleton-code amendments and mixed Paper 1/Paper 2 revision. Guided reading task: Read skeleton-code documentation or code comments and summarise the program flow.
13 Term 3	Revision and reinforcement	- Complete synoptic retrieval and timed practice across the specification. - Refine examination technique, mathematical working and extended responses. - Correct misconceptions using mark schemes and examiner-style feedback.	- All A Level content is interleaved. - Focus determined by mock and retrieval evidence.	- Full or adapted past papers. - Diagnostic quizzes and targeted question sets. - Final examination-readiness checks.	- Personalised revision priorities. - Tiered support and stretch tasks. - Accessible summaries, diagrams and model answers with scaffolding gradually removed.	Do now tasks: Interleaved recall, calculation practice, short code trace or exam-command-word starter. Classwork tasks- written: Complete timed papers, mark responses and produce corrected model answers. Classwork tasks- independent research: Use assessment analysis to build a personalised final revision plan. Homework tasks: Targeted exam-question practice from priority topics. Guided reading task: Read examiner commentary and annotate what distinguishes mid- and high-level answers.