The Fisher Way: Curriculum

The Fisher Way aims to educate and inspire with joy, faith and love because we are an inclusive Catholic community.

Successful and resilient learners who aspire to and achieve excellence

Confident individuals who can explore and communicate effectively

Responsible citizens who are active, loving and wise in all their endeavours

Subject	A Level Computer Science
Year Group	Year 12
Intent	**Successful and Resilient Learners:** who can analyse problems and find efficient, creative solutions for both real world situations and artificial systems **Confident Individuals**: who can use digital technology effectively, creatively and safely in their personal lives and future careers **Responsible Citizens**: who understand the social and cultural impact that technology has on their lives and the lives of others.

Narrative	The Year 12 Computer Science curriculum follows the AQA A Level Computer Science specification. Theory and programming are taught side by side, building the knowledge needed for Paper 1, the on screen programming and problem solving exam, and Paper 2, the written theory exam, as well as the non exam assessment. In the autumn term, theory covers data representation, from number systems and binary arithmetic to images, sound, compression and encryption, followed by computer systems, logic gates and Boolean algebra, and the hardware and software that make up a computer, including processor architecture and assembly language. In programming, students secure the fundamentals in Python, including variables, data types, selection, iteration, arrays, subroutines, files, events and graphical user interfaces, before being introduced to C#. In the spring term, theory moves to networks and the Internet. In programming, students develop validation and more advanced selection and iteration techniques, then move on to object oriented programming, recursion and data structures such as stacks, queues, graphs and trees. In the summer term, students study the social, ethical, legal and cultural consequences of computing. In programming, they learn the standard algorithms in the specification and begin their NEA programming project, identifying a problem and a client and starting their analysis. By the end of Year 12, students have a secure foundation in both papers and a clear direction for their NEA in Year 13.					
Half term	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Knowledge (topics studied)	Theory: Data Representation Theory: Computer Systems Theory: Hardware and Software Programming: Python Fundamentals Programming: Introducing C#		Theory: Networks and The Internet Programming: Validation Programming: Advanced Selection and Iteration Programming: Object Orientation Programming: Recursion Programming: Data Structures		Theory: Social and Ethical Considerations Programming: Algorithms Programming: NEA Programming Project	
Key skills	Students work with number bases, two's complement, fixed point binary and binary arithmetic, and explain how characters, images and sound are represented, including error checking, compression and encryption. They classify software and programming languages, explain how code is translated, and use logic gates and Boolean algebra to simplify expressions. They describe processor architecture, the fetch execute cycle, addressing modes and assembly		Students compare serial and parallel, synchronous and asynchronous transmission, network topologies, and client server and peer to peer models. They explain the TCP/IP stack, IP addressing and subnetting, DNS, firewalls, encryption and digital certificates, and how web applications use client server communication. In programming, they validate input and handle exceptions, use more advanced selection and iteration, and design programs with classes, objects, inheritance, encapsulation		Students discuss the moral, ethical, legal and cultural issues raised by computing, including privacy, surveillance, artificial intelligence and the responsibilities of developers, and apply relevant legislation to scenarios. In programming, they trace and implement graph and tree traversals, Reverse Polish notation, searching and sorting algorithms including binary search and merge sort, and Dijkstra's shortest path algorithm. They begin their NEA by	

	language, and the factors that affect performance. In programming, they write Python programs using variables, data types, selection, iteration, arrays, subroutines and files, build event driven programs with a graphical user interface, and then begin writing console applications in C#.	and polymorphism. They write recursive subroutines and implement data structures including stacks, queues, graphs, trees, hash tables and dictionaries.	choosing a problem, identifying a client or end user, researching existing solutions and starting their analysis with measurable objectives.
Cultural capital	Students gain a deep understanding of how computers work, from a single logic gate to a complete processor running an operating system. Learning a second, strongly typed language shows them how professional developers move between languages, and prepares them for the C# skeleton program used in Paper 1.	Students understand how the Internet actually works and how it is secured, giving them an informed view of the technology behind every website and app. Object oriented programming introduces the design approach used across the software industry, preparing them for larger, team based projects.	Students consider the power and responsibility that come with building technology that affects millions of people. Starting an independent project with a real client mirrors the way professional developers work, building the independence, planning and communication skills valued at university and in industry.
Assessment	End of Unit Assessments Programming Tasks	End of Unit Assessments Programming Tasks Mock Exams	End of Unit Assessments NEA Analysis