Text

Description automatically generated with medium confidence

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sir Harry Smith Community College Curriculum Map SUBJECT: Computer Science YEAR 11 2022-23** | | | | |
| **Curriculum Intent:** To provide students with a challenging and engaging computer science curriculum, that builds on prior learning at KS3 and Whittlesey’s local context. To develop further knowledge and understanding of computer science fundamentals and build fluency in writing coded solutions which solve real-world problems. | | | | |
| **School Values** | **Curriculum Focus** | **Term 1 – Algorithms** | **Term 2 – Programming Fundamentals** | **Term 3 – Producing Robust Programs and Language Development** |
| **High Quality Learning Experience** | **Literacy Skills and Key Vocabulary** | Abstraction, Decomposition, Algorithm, Pattern recognition, Input, Output, Process, Sub-program, Terminal, Decision, Line, Syntax, Logic, Binary, Linear, Bubble, Merge, Insertion, Pseudocode, Flowchart, Divide and Conquer, Pass, Iterate, Instruction, Sequence, Selection, Iteration. | Instruction, Sequence, Selection, Iteration, Comparison operator, Boolean operator (AND, OR, NOT), Arithmetic operators, Comment, Assignment, Constants, Global variable, Casting, Iteration (count and condition controlled loops), Selection (If-Then-Else), String handling, Concatenate, Uppercase, Lowercase, Substring, ASCII, Declaration, Procedure, Function, Argument, Parameter. | Input sanitisation, Input validation, Defensive design, Authentication, Maintainability, SQL injection, Whitelist, Blacklist, Indentation, Formatting, Comments, Modularised, Iterative testing, Black box testing, White box testing, Normal, Boundary, Invalid, Alpha, Beta, Compiler, Interpreter, IDE, Machine code, Op code, Operand, Mnemonic, Assembler, Source code, Automation tools, Debugger. |
| **Pursuit of Excellence** | **Knowledge and Skills** | **Knowledge of computational thinking techniques**  Abstraction.  Decomposition.  Algorithmic thinking.  **Knowledge of how to design, create and refine algorithms**  Identify the inputs, processes, and outputs for a problem.  Structure diagrams.  Create, interpret, correct, complete, and refine algorithms using pseudocode, flowcharts, reference language.  Identify common errors.  Trace tables.  **Knowledge of searching and sorting algorithms**  Binary search.  Linear search.  Bubble sort.  Merge sort.  Insertion sort. | **Knowledge of programming fundamentals**  The use of variables, constants, operators, inputs, outputs, and assignments.  The use of the three basic programming constructs used to control the flow of a program (sequence, selection, and iteration).  The common arithmetic operators.  The common Boolean operators AND, OR, and NOT.  **Knowledge of Boolean logic**  Simple logic diagrams using the operators AND, OR, and NOT.  Truth tables.  Combining Boolean operators using AND, OR, and NOT.  Applying logical operators in truth tables to solve problems.  **Knowledge of data types**  The use of the following data types: integer, real, Boolean, character, string, and casting.  **Knowledge of additional programming techniques**  The use of basic string manipulation.  The use of basic file handling operations (open, read, write, close).  The use of records to store data.  The use of SQL to search for data.  The use of arrays when solving problems, including both one-dimensional (1D) and two-dimensional (2D) arrays.  How to use sub programs (functions and procedures) to produce structured code.  Random number generation. | **Knowledge of defensive design strategies**  Defensive design considerations (anticipating misuse, authentication).  Input validation.  Maintainability (sub programs, naming conventions, indentation, commenting).  **Knowledge of testing**  Purpose of testing.  Types of testing (iterative and final).  Identify syntax and logic errors.  Selecting and using suitable test data (normal, boundary and invalid).  Refining algorithms.  **Knowledge of programming languages**  Characteristics and purpose of different levels of programming language (high-level and low-level).  Purpose of translators.  The characteristics of a compiler and an interpreter.  **Knowledge of Integrated Development Environment’s (IDEs)**  Common tools and facilities available in an IDE (editors, error diagnosis, run-time environment and translators). |
| **Subject specific pedagogy** | Lead with concepts – support students in the acquisition of knowledge through use of key concepts, terms and vocabulary, providing opportunities to build a shared and consistent understanding.  Work together – collaboration is encouraged, with peer instruction and feedback. Working together stimulates dialogue, articulation of concepts and development of understanding.  Unplug, unpack, repack – new concepts introduced by unpacking complex terms and ideas in familiar contexts, then repacking new understanding into the original concept.  Add variety – activities with different levels of direction, scaffolding, and support to promote learning, ranging from highly structured to more exploratory tasks.  Thinking aloud – thinking verbally out loud to model thought process to learners.  Language and context – ensuring subject matter is contextually and culturally relevant to all students, and language/literacy is accessible and understandable for all learners. | Worked examples – demonstration of ‘expert’ solutions to a problem.  Live coding – developing programming solutions in real time.  Code tracing – reading and interpreting the code, and recording the flow and/or outputs.  Modelling for learners – making deliberate mistakes to enable learners to fully understand how the code behaves.  Code maintainability – use of appropriate variable names and conventions, code chunking / modularised code and comments throughout the code.  T.I.M.E – Try; Investigate; Make; Evaluate.  Try (predict what a program will do, then type the code and run to check).  Investigate (comprehend the code, modify and debug the code).  Make (design an algorithm and code a solution).  Evaluate (test the code, reflect on the approach, refine the code). | |
| **Extending the boundaries of learning** | **Cultural Capital and beyond the curriculum** | Students will have the opportunity to attend an enrichment computing club to allow them opportunities to develop their knowledge, understanding and skills beyond the normal taught curriculum.  All students will be enrolled onto the Inspiring Digital Enterprise Award (iDEA), where they will complete a series of challenges and learn vital skills for work and life.  Students will have an opportunity to learn about John Von Neumann who was a Hungarian-American Mathematician, Physicist and Computer Scientist who invented the Merge sort algorithm which is an efficient, general purpose, comparison-based sorting algorithm using the divide-and-conquer technique. | Students will have the opportunity to attend an enrichment computing club to allow them opportunities to develop their knowledge, understanding and skills beyond the normal taught curriculum.  All students will be enrolled onto the Inspiring Digital Enterprise Award (iDEA), where they will complete a series of challenges and learn vital skills for work and life.  Students will build on prior learning from KS3 and explore further the works of George Boole, who was the First Professor of mathematics at Queen’s College, Cork, in Ireland. He worked in the field of algebraic logic and wrote the Laws of Thought (1854) which contains Boolean algebra, laying the foundations for the Information Age. | Students will have the opportunity to attend an enrichment computing club to allow them opportunities to develop their knowledge, understanding and skills beyond the normal taught curriculum.  All students will be enrolled onto the Inspiring Digital Enterprise Award (iDEA), where they will complete a series of challenges and learn vital skills for work and life.  Students will have an opportunity to look at a number of case studies where organisations have been hacked or suffered severe data loss due to disregard for the development and implementation of robust computer systems within society. Students will consider the moral, ethical, legal and cultural impact within their local community and wider society. |
| **Achievement** | **Assessment** | Throughout the unit of work, students will receive regular oral and written feedback to address any misconceptions and enable them to move their learning forwards.  Opportunities for self and peer-assessment are embedded throughout each lesson.  Each lesson includes a formative assessment opportunity to address misconceptions from the learning.  Students complete a summative assessment at the end of the unit. Once the assessment has been marked, students will be provided with feedback to enable them to close gaps in learning from the unit. | | |
| **Valuing People** | **How our curriculum meets the needs of every individual** | All students are issued with a BBC Micro:bit upon starting the course, to enable them to work independently and freely on programming projects of their choice.  The computing curriculum builds on students’ prior knowledge, understanding and skills, and considers local context of Whittlesey to produce a challenging and engaging computing curriculum.  The computing department works closely alongside the Supportive Skills Department (SSD) to provide personalised support to those students that require it.  Students are encouraged to work at their own pace in lessons, taking the time they need to securely embed learning, we therefore place all teaching resources on the class Team so that students can go back over existing or new materials from the lesson.  Teaching staff are aware of prior attainment levels, and this is built into lesson planning the sequencing of the curriculum.  Students are supported with learning new vocabulary and developing their oracy to promote and deepen their knowledge and understanding of subject matter.  Computing department works closely with the pastoral team to develop a holistic understanding of all learners.  Students are provided with opportunities and encouraged to be independent and resilient learners, so that they are equipped and able to take ownership over their learning and progress.  Adaptive teaching to provide support to students as and when needed in order to move their learning forwards and close any gaps. | | |