**Learning Programme**

**Fundamentals of Algorithms – GCSE**

| Topic/Content | Objectives/Skills | Homework | Assessment | Stretch & Challenge (Thirst for Learning) |
|---|---|---|---|---|
| Data Types | • Understand the concept of a data type.<br>• Understand and use the following appropriately:<br>  ○ Integer<br>  ○ Real<br>  ○ Boolean<br>  ○ Character<br>  ○ String. | | Students will complete a number of different programming tasks<br><br>Use random numbers in programming<br><br>Students will implement the methods taught in their programming projects<br><br>Q and A in lessons<br><br>Theory questioning worksheets<br><br>End of topic test | Students will make use of the correct data types when programming and ensure the most suitable data type for the situation is used.<br><br>Students will learn how to implement and make use of functions within their programming |
| Programming Concepts | • Use, understand and know how the following statement types can be combined in programs:<br>  ○ variable declaration<br>  ○ constant declaration<br>  ○ assignment<br>  ○ iteration<br>  ○ selection<br>  ○ subroutine (procedure/function).<br>• Use definite and indefinite iteration, including indefinite iteration with the condition(s) at the start or the end of the iterative structure.<br>• Use nested selection and nested iteration structures.<br>• Use meaningful identifier names and know why it is important to use them. | | | |

| | | | | |
|---|---|---|---|---|
| Arithmetic operations in a programming language | • Be familiar with and be able to use:<br>  ○ Addition<br>  ○ subtraction<br>  ○ multiplication<br>  ○ real division<br>  ○ integer division, including remainders. | | | |
| Relational operations | • Be familiar with and be able to use:<br>  ○ equal to<br>  ○ not equal to<br>  ○ less than<br>  ○ greater than<br>  ○ less than or equal to<br>  ○ greater than or equal to. | Complete research into the different types of Boolean Operations | | |
| Boolean Operations | • Be familiar with and be able to use:<br>  ○ NOT<br>  ○ AND<br>  ○ OR. | | | |
| Data Structures | • Understand the concept of data structures.<br>• Use arrays (or equivalent) in the design of solutions to simple problems.<br>• Use records (or equivalent) in the design of solutions to simple problems. | | | |
| Input/output and file handling | • Be able to obtain user input from the keyboard.<br>• Be able to output data and information from a program to the computer display.<br>• Be able to read/write from/to a text file. | | | |

| | | | | |
|---|---|---|---|---|
| String handling operations | • Understand and be able to use:<br>  ○ Length<br>  ○ Position<br>  ○ Substring<br>  ○ Concatenation<br>  ○ convert character to character code<br>  ○ convert character code to character<br>  ○ string conversion operations. | | | |
| Random number generation | • Be able to use random number generation. | | | |
| Subroutines (procedures and functions | • Understand the concept of subroutines.<br>• Explain the advantages of using subroutines in programs.<br>• Describe the use of parameters to pass data within programs.<br>• Use subroutines that return values to the calling routine.<br>• Know that subroutines may declare their own variables, called local variables, and that local variables usually:<br>  ○ only exist while the subroutine is executing<br>  ○ are only accessible within the subroutine.<br>• Use local variables and explain why it is good practice to do so. | | | |
| Structured programming | • Describe the structured approach to programming. | | | |

| | | | | |
|---|---|---|---|---|
| | • Explain the advantages of the structured approach | | | |
| Robust and secure programming | • Be able to write simple data validation routines.<br>• Be able to write simple authentication routines.<br>• Be able to select suitable test data that covers normal (typical), boundary (extreme) and erroneous data.<br>• Be able to justify the choice of test data. | | | |
| Classification of programming languages | • Know that there are different levels of programming language:<br>  ○ low-level language<br>  ○ high-level language<br>• Explain the main differences between low-level and high-level languages.<br>• Know that machine code and assembly language are considered to be low-level languages and explain the differences between them.<br>• Understand that ultimately all programming code written in high-level or assembly languages must be translated into machine code.<br>• Understand that machine code is expressed in binary and is specific to a processor or family of processors.<br>• Understand the advantages and disadvantages of low-level language programming compared with high-level language programming. | Homework<br>Worksheet | | |

| | | | | |
|---|---|---|---|---|
| | • Understand that there are three common types of program translator:<br>  ○ interpreter<br>  ○ compiler<br>  ○ assembler<br>• Explain the main differences between these three types of translator.<br>• Understand when it would be appropriate to use each type of translator. | | | |