**Learning Programme**

**Fundamentals of programming – AS Level**

| Topic/Content | Objectives/Skills | Homework | Assessment | Success Criteria | Stretch & Challenge (Thirst for Learning) |
|---|---|---|---|---|---|
| Programming | | | Students have both class mock tests and the official mock test on paper 1. This is a live programming exam with the students answering questions on the computer based on the pre-release they have studies. | Grades can be based on the actual grade boundaries from the previous year's exam | Students are encouraged to start learning the programming language C#, which the majority of students who continue to the Upper 6th use to create their A' level project. |
| Data types | • Understand the concept of a data type.<br>• Understand and use the following appropriately:<br>  ○ integer<br>  ○ real/float<br>  ○ Boolean<br>  ○ character<br>  ○ string<br>  ○ date/time<br>  ○ records (or equivalent)<br>  ○ arrays (or equivalent).<br>• Define and use user-defined data types based on language-defined (built-in) data types. | Students will be given an AS and A'level paper 1 pre-release throughout the year and will have to study and make amendments to the program as they see it. (solutions are guess work, but a few can be accurately guessed) | | | |
| 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 | | | | |

| | | | | | |
|---|---|---|---|---|---|
| | ○ 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. A theoretical understanding of condition(s) at either end of an iterative structure is required, regardless of whether they are supported by the language being used.<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/float division<br>○ integer division, including remainders<br>○ exponentiation<br>○ rounding<br>○ truncation | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Relational operations in a programming language | • 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 | | | | |
| Boolean operations in a programming language | • Be familiar with and be able to use:<br>  ○ NOT<br>  ○ AND<br>  ○ OR<br>  ○ XOR | | | | |
| Constants and variables in a programming language | • Be able to explain the differences between a variable and a constant.<br>• Be able to explain the advantages of using named constants. | | | | |
| String-handling operations in a programming language | • Be familiar with and be able to use:<br>  ○ length<br>  ○ position<br>  ○ substring<br>  ○ concatenation<br>  ○ character → character code<br>  ○ character code → character<br>  ○ string conversion operations. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Random number generation in a programming language | • Be familiar with, and be able to use, random number generation | | | | |
| Exception handling | • Be familiar with the concept of exception handling<br>• Know how to use exception handling in a programming language with which students are familiar. | | | | |
| Subroutines (procedures/functions) | • Be familiar with subroutines and their uses.<br>• Know that a subroutine is a named 'out of line' block of code that may be executed (called) by simply writing its name in a program statement.<br>• Be able to explain the advantages of using subroutines in programs. | | | | |
| Parameters of subroutines | • Be able to describe the use of parameters to pass data within programs.<br>• Be able to use subroutines with interfaces. | | | | |
| Returning a value/values from a subroutine | • Be able to use subroutines that return values to the calling routine. | | | | |
| Local variables in subroutines | • Know that subroutines may declare their own variables, | | | | |

| | | | | | |
|---|---|---|---|---|---|
| | called local variables, and that local variables:<br>○ exist only while the subroutine is executing<br>○ are accessible only within the subroutine.<br>• Be able to use local variables and explain why it is good practice to do so. | | | | |
| Global variables in a programming language | • Be able to contrast local variables with global variables. | | | | |
| **Structured programming** | | | | | |
| Structured programming | • Understand the structured approach to program design and construction.<br>• Be able to construct and use hierarchy charts when designing programs.<br>• Be able to explain the advantages of the structured approach. | | | | |