Saint Martin's Catholic Academy

Computer Science - Year 7 Long Term Plan

Time allocation	1 lesson per week		
Unit 2 – Programming using Small Basic			
Key learning in this unit: Students learn that computers are binary devices, controlled by switches called transistors. They briefly study the history of computing to understand how and why we have arrived at this point and why binary devices are more reliable than those which might try to represent decimal quantities electronically. The programming language "Microsoft Small Basic" is then used to introduce students to a text based, object-oriented style programming language where they gain familiarity with the core building blocks of any program – variables, input/output, iteration and decision making. Small Basic is an ideal first programming language as it has only 13 key words, yet is very powerful. The language prevents common misconceptions that arise when using more "visual" learning tools such as Scratch which can quickly become disconnected from the actual skills necessary to produce code in the real world.			
	Key learning, tasks and skills	**Opportunities for assessment**	**Specific knowledge focus**
Lesson 1 – Computers and Algorithms	• Students learn the basic terminology necessary to understand what a computer is and how it might be controlled using a programming language – binary, transistor, input, process, output, algorithm. • Students learn to draw and interpret flow charts to represent and understand simple algorithms such as creating a username, resetting a password or deciding the outcome of a game. • Students learn the core concepts behind input, output and variables in a flow chart.	• Knowledge check – what is a computer, basic algorithm facts. • Tasks completed in lesson – flow charts, written answers to tasks.	• A computer is a binary device – it understands data in the form of the numbers 1 and 0 only. • Computers represent data, logic and instructions using switches called transistors. These may be on or off (1 or 0) • Computers take input (data going IN to a machine), process it according to program instructions in order to produce output (data coming OUT of a machine) • Programs/software are a list of mathematical or logical instructions. • Algorithms are specific sequences of instructions which solve a particular problem.

Lesson 2 - Variables	• Through a process of programming examples which build on each previous example, students explore the concept of adding variables to their programs and observing the effect of changing the value of variables on each run of a program. • Students learn the purpose and advantages of variables. • Students learn to create variables which store both numeric and text data • Students learn to combine variables and manipulate their contents using operators such as + for both addition and concatenation	• Knowledge Check – Key terms – algorithms, input, process, output, variable • Program code produced, feedback from errors and debugging process. • Answers to written tasks	• Variables are based on the concept of algebra – an identifier which represents a value. • Variables are an extension of algebra – we may use clear, long names as identifiers rather than single letters that are commonly used in Maths. • The contents of variables may change, this is why a program which uses them may produce different output each time it is executed. • Variables allow programs to be changed quickly, by simply adjusting the value of a variable in one place – which will then cascade those changes anywhere else it is used.
Lesson 3 – Input / Output	• Students develop the concept of using variables and concatenation to explore taking and storing input to produce meaningful and well formatted output. • Students develop their knowledge of variables that store numeric data to produce a number of programs which apply various formulae to inputs in order to provide useful output using mathematical operators	• Knowledge Check – Variables • Program code produced, feedback from errors and debugging process. • Answers to written tasks	• Input is the process of taking data from the outside world (user, sensor, data stream) and storing it in a variable so we can use it in a program. • Input allows programs and their output to be tailored to the specific requirements of a user. • Output should be sensibly formatted and concatenation is the key to this • Concatenation allows us to insert the contents of variables as a form of “find and replace” or “fill in the blanks” in output.
Lesson 4 - Loops	• Students are introduced to the idea that programs are repetitive in nature and this leads to them being significantly inefficient without the use of loops. Loops build on their previous learning involving variables and, through examples, students discover that variables can be used not only to control a piece of code	• Knowledge Check – Input, output and concatenation • Program code produced, feedback from errors and debugging process. • Answers to written tasks	• A loop allows a block of code to be repeated either a set number of times or until the conditions of a rule are met. • There are two main types of loop – Fixed (FOR) loops and condition controlled (WHILE) loops.

	being executed repeatedly, but also to be used as useful output during this process. • Students create programs involving fixed loops or “FOR… NEXT” loops and, through a process of gradual development, learn to use the loop counter as an important variable in their programs. • Students finally learn about condition controlled or “WHILE” loops and learn to build a program which iterates an unknown amount of times, based on the input which the program receives from the end user.		• FOR loops are used when you need to repeat a section of code a specific number of times. • WHILE loops are used when you do not know how often or long something will need to happen for. For example, people playing a game may have vastly different skill levels – WHILE loops allow for players to continue UNTIL they lose the game.
Lesson 5 – Arrays	• This lesson combines all prior learning of input, output, variables and loops to introduce the concept of an array, which is effectively a table of similar data stored in a program. • Students learn, through a process of gradual evolution, to create a program which stores a collection of data entered by the user. • Students learn to iterate through an array using a loop and the importance of the loop counter in doing so. They learn to systematically manipulate each element in the array on each iteration of a loop.	• Knowledge Check – Loops • Program code produced, feedback from errors and debugging process. • Answers to written tasks	• An array is a collection of data, much like a table • Arrays are fixed size • Arrays contain data all of the same type – e.g. all text or all numeric. • Arrays are indexed – each “row” has a number which we use to refer to the data stored there. • Loops are *essential* prior learning as you cannot sensibly use an array without a loop to fill, search, edit or otherwise manipulate that array
Lesson 6 – If’s	• This lesson introduces the idea of changing the flow of program execution based on the outcome of a particular rule. • Students learn to create programs which give different output based on a rule applied to the contents of a variable.	• Knowledge Check – Arrays • Program code produced, feedback from errors and debugging process. • Answers to written tasks	• Programs often need to provide different actions or outputs depending on the outcome of a condition. • IF statements allow programs to execute different blocks of code according to the result of a calculation

	• Students learn how to extend their programs to test for multiple conditions and therefore provide multiple possible outcomes using the "else if" and catch-all "else" statements.		• "ELSE" is the catch all clause which allows programmers to specify what happens if none of the previous conditions are met. • IF statements may have multiple tests or conditions and multiple possible outcomes. For example – a menu may have 15 options, all of which need a different section of code to consequentially be executed.
Lesson 7 – Debugging	• In their final lesson, students learn about the methods for identifying and systematically removing bugs from their programs. • Students are introduced to the two main types of program error – syntax and logic. • Students learn to interpret and understand common error messages and how these can lead to the rapid resolution of program errors.	• Knowledge Check – If's • Program code produced, feedback from errors and debugging process. • Answers to written tasks	• All software contains errors • Syntax errors are often spelling mistakes or misunderstandings of how the grammar of a programming language should be applied • Syntax errors are simple to find and fix as the program will refuse to run until they are corrected • Logic errors are mistakes a programmer made in their thinking or application of an algorithm • They are much more difficult to find and fix as programs will run, but will give unexpected output.