Repetition – *Repeating a set of steps several times.*

Count controlled:
- Repeats the same code a set number of times
- Uses a variable to track how many times the code has been run
- This variable can be used in the loop
- At the end of each iteration the variable is checked to see if the code should be run again
- FOR sets how many times the code should be repeated
- NEXT tells the code to return to the start of the loop
- STEP sets how the variable should increment

Condition Controlled:
- Uses a condition to determine how many times code should be repeated
- While loops will run whilst a condition is met and use the statements WHILE and ENDWHILE
- Repeat loops will run until a condition is met and use the statements REPEAT and UNTIL

Evaluating Fitness for Purpose and Efficiency

Fit for Purpose - meets the original purpose and requirements the code was designed for. Provides the expected outputs. Test tables help to examine the values at each stage and check code is working as expected.
Efficient – the amount of time and resources needed to run a particular program.
Steps which improve efficiency:
- Using repetition (loops) to reduce the amount of code
- Using arrays instead of declaring many individual variables
- Using selection statements which only make comparisons until a solution is reached

Abstraction - *Using symbols and variables to represent a real-world problem using a computer program and removing unnecessary elements*

Advantages:
- Allows the creation of a general idea of how to solve the problem.
- Provides focus on what actually needs to be done.
- Provides a simple view of the problem

Variables - *a box in which data may be stored*

- Different types e.g. string, decimal, etc.
- Allows the program to store data such as an input for later use

Pseudocode

- Uses short English words and statements to describe an algorithm.
- Generally, looks a little more structured than normal English sentences.
- Flexible.
- Less precise than a programming language.

Flowcharts

- Created to represent an algorithm.
- Show the data that is input, and output.
- Show processes that take place.
- Show any decisions and repetitions that take place.
- Lines show flow through the chart.
- Shapes represent different functions

Arrays - *An ordered collection of related data*

- Each element in the array has a unique index, usually starting at 0
- All elements must be the same type of data
- Arrays are usually a fixed size
- **1 Dimensional arrays** are like a simple list, each element needs a single index number
- **2 Dimensional arrays** are like tables, with each element needing two index numbers
- 2 Dimensional arrays are usually used to store properties of objects, with objects in rows and properties in columns
- Fruits[1] references element 1 in the 1D Fruits array
- Tools[0,2] references element 0,2 in the Tools array

Topic 1 – Computational Thinking

Operators – *Allow us to work with data, to change it and compare it*

Arithmetic Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
MOD	Modulus (the remainder from a division, e.g. 12 MOD 5 gives 2)
DIV	Quotient (integer division, e.g. 21 DIV 5 gives 4)
^	Exponentiation (to the power of, e.g. 3^3 gives 27)

Comparison Operators

==	Equal to
!=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Boolean Operators

AND - two conditions must be met for the statement to be true
OR - at least one condition must be met for the statement to be true
NOT – inverts the result, e.g. NOT(A AND B) will only be false when both A and B are true

Sequencing - *Breaking down complex tasks into simple steps.*

- The order of steps matter
- Step by step progress through a program
- Benefits
  - Each line follows the next.
  - Can create simple programs very quickly.
  - Easy to follow for a small program.
- Disadvantages
  - Not very efficient.
  - Difficult to follow with large programs.
  - Hard to maintain.

Types of Error

A program with a syntax error will not run. A program with a logic error will run but it will not perform as expected.

Syntax Errors
When the code does not follow the syntax rules of the programming language used. This stops the program from running.
Examples:
- Misspellings or typos
- Using a variable before it has been declared
- Missing or incorrect use of brackets

Logic Errors
The program runs but does not do what it should.
Examples:
- Incorrectly using logical or Boolean operators
- Creating infinite loops
- Incorrectly using brackets in calculations
- Using the same variable name at different points for different purposes

Runtime Errors
Takes place during the running of a program causing it to crash.
- Trying to divide by zero
- Trying to access item 6 in an array of 5 items

Sorting Algorithms

Bubble Sort
1) Take the first element and second element
2) Compare the two
   a) If element 1 > element 2
      i) Swap them over
   b) Otherwise
      i) Do nothing
   c) Move to the next pair in the list
   d) If there are no more elements return to step (1)
   e) Otherwise, return to step (2)
3) Repeat until you have worked through the whole list without making any changes

Merge Sort
1) Split the list into individual elements.
2) Merge the elements together in pairs, putting the smallest element first.
3) Merge two pairs together, putting the smallest first.
4) Keep merging until all pairs are in order.

Searching Algorithms

Linear Search
1) Check the first value
2) If it is desired value
   a) Stop
3) Otherwise check the second value
4) Keep Going until all elements have been checked or the value is found

Binary Search
1) Put the list in order.
2) Take the middle value.
3) Compare it to the desired value.
   a) If it is the desired value.
      i) Stop.
   b) If it is larger than the desired value.
      i) Take the list to the left of the middle value.
   c) If it is smaller than the desired value.
      i) Take the list to the right of the middle value.
4) Repeat step 3 with the new list.

Sub Programs - *Small programs which form part of a larger program.*

Procedures are sets of instructions stored under a single name (identifier).
Functions are like procedures but will always return a value to the main program.
Parameters are values passed into a sub program. These are referred to as arguments when calling the sub program.
Advantages:
- Used to save time and simplify code
- Allows the same code to be used several times without having to write it out each time
- Usually small in size, so easier to write and test.
- Easy for someone else to understand.
- Can be saved separately as modules and used again in other programs.
- Saves time because code that has already been written and tested can be reused

Trace Tables – *a method of recording the values used within an algorithm at each stage of processing to help in troubleshooting*

- Tests algorithms for logic errors which occur when the algorithm is executed.
- Simulates the steps of algorithm.
- Each stage is executed individually allowing inputs, outputs, variables, and processes to be checked for the correct value at each stage.
- A great way to spot errors

Decomposition - *Breaking down large problems into a set of smaller parts.*

There are several different approaches, and not one single right way to do this.
Advantages:
- Smaller problems are easier to solve
- Each part can be solved independently
- Each part can be tested independently
- The parts are combined to produce the full problem.
- Allows each smaller problem to be examined in more detail

Selection

- Allows the program to make decisions
- Uses conditions to change the flow of the program
- Selections may be nested one inside another
- IF statements perform comparisons sequentially and so the order is important
- SELECT CASE has less typing but is less flexible

Constants – *a fixed value used by the program such as pi*

Allows easy use of fixed values without having to store them in the program