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.

Data Types

- Integers – whole numbers e.g. 27
- Reals – numbers containing decimals e.g. 56.2
- Boolean – TRUE or FALSE
- Character – An alphanumeric character e.g. a
- Strings – One or more alphanumeric characters e.g. hello

Subroutines (Procedures and functions)

- Used to save time and simplify code
- Allows the same code to be used several times without having to write it out each time
- Procedures are sets of instructions stored under a single name
- Functions 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
- Both procedures and functions can accept parameters
- The subroutine can return an output back to the main program to allows the result of the subroutine to be used in the main program.
- This is done using the Output 'return value' code in a subroutine.
- Variables created within a subroutine are called **Local Variables**, they only exist whilst the subroutine is running.
- They are only accessible to that subroutine and not to other subroutines or the main program.
- This makes the code easier to debug.
- This improves efficiency since the variable does not exist when the subroutine is not running, freeing up memory.
- **Global variables** can be used anywhere within the code.
- Because they can be altered anywhere in the program they are harder to troubleshoot.

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
- 1D arrays are similar to a simple list, each element needs a single index number
- 2D arrays are similar to tables, with each element needing two index numbers
- 2D 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

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

Unit 2 - Programming

String Manipulation

- stringname.length – returns the length of a string
- stringname.upper – converts the string to uppercase

`string = "John"`		
`string.length`	The length of the string	4
`string.upper`	Convers to upper case	JOHN
`string.lower`	Converts to lower case	john
`string.substring(1,2)`	Returns part of the string	oh
`string.left(3)`	Returns from the left of the string	Joh
`string.right(2)`	Returns from the right hand side of the string	hn
`string+string`	Concatenates or joins strings	JohnJohn

Keywords

Variables:

- A box in which data may be stored
- Content changes as the program runs.
- Different types e.g. string, decimal, etc.

Assignment:

- The process for changing the data stored in a variable
- Copies a value into a memory location
- Different values may be assigned to a variable at different times during the execution of a program.
- Each assignment overwrites the current value with a new one.

Constants:

- Data does not change as the program runs
- Used to reference known values such as pi

Inputs:

- May come from the user, a file or elsewhere in a modular program
- Usually treated as text even if containing numbers

Outputs:

- The end result of the program
- May be displayed on the screen, written to a file, or sent to a device

Operators:

- Used to manipulate and compare data

Operators

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

Testing

- Newly written programs often contains bugs which stop them working properly.
- Testing allows the programmer to locate and remove these bugs, making sure the program meets its' needs.
- Different types of data will need to be entered into the program to test it is working correctly.
- This data should cover a range of different data which the program might have to deal.
- It should cover possible data, which is normal and allowed, and impossible, which is not allowed.
- **Normal Data** – is typical data that the program is likely to encounter and should be able to process without error.
- **Boundary Data** – is data at the limit of what the program will and will accept.
- **Erroneous Data** – is data which is not valid.

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

Data Validation

- Users may enter incorrect data.
- Computer programs should be able to check for this and take appropriate action.
- Validation applies rules to data, only data which meets the rules is allowed.
- This reduces the risk that an incorrect input will crash the program.
- Validation does not ensure that the data is correct, but that it is in the correct format.
- We can use comparison and string manipulation operators combined with selection and subroutines to implement validation.
- **Range Check** – makes sure that the data is within a certain range, this is usually used for numbers and dates
- **Length Check** – makes sure the data is the correct length, and not too long or short.
- **Presence Check** - makes sure a value has actually been entered. For example, an email address must be entered to sign up for a newsletter.
- **Format Check** – makes sure the data is in the correct format.
- **Type Check** – Makes sure the data is of the correct data type

Authentication

- Confirms the user is who they say they are
- Commonly accomplished by asking the user to enter a username and password.
- The username and password are stored persistently, such as in a text file.

Iteration

- Running through or 'iterating' through a set of steps several times.
- Also known as looping
- Count Controlled (Definite) iteration
  - 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 determine 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 (Indefinite) Iteration
  - 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

```
FOR count = 2 to 10 STEP 2
      OUTPUT count * 3
NEXT count

count = 0
WHILE count < 6
     print("Hello World")
     count = count + 1
ENDWHILE
```

Random Numbers

- Many different applications in computer programs from simulating dice in computer games, to cryptography
- Depending on the language we may specify just the maximum number assuming starting from 1 (e.g. roll = random(5)) or the first and last possible values (e.g. roll = (3,9))
- In many cases our desired output may not be a number and so we must then use selection, such as an IF or CASE statement, to convert the number into an actual choice
- We can also use the random number to select a random element from an array. This is more efficient then writing lots of IF statements.