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
- Strings – alphanumeric characters e.g. hello
- Casting is used to convert data from one type to another. This is often used to convert string input to integer or real to allow for calculation

Sub Programs

- 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 (identifier)
- Functions are similar to 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
- Both procedures and functions can accept parameters

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

The Use Of Records To Store Data And SQL

- Data is often stored in databases, providing persistent storage for data.
- Data within databases is stored in records, which in turn are stored in files.
- Records contain several attributes, each attribute is a single point of data.
- SQL (Structured Query Language) is a programming language designed for interacting with databases.
- SQL uses the SELECT command to search and read databases

SELECT * FROM Books	Returns all columns and records in the Books table
SELECT Title FROM Books	Returns only the title column from the Books table
SELECT * FROM Books WHERE Author=”Bob”	Searches the Books table for records where the Author is Bob. Returns all Columns
SELECT * FROM Books WHERE Author=”Bob” or Author=”Tim”	Searches the Books table for records where the Author is Bob or Tim. Returns all Columns
SELECT * FROM Books WHERE Author!=”Bob”	Searches the Books table for records where the Author is not Bob. Returns all Columns
SELECT Title FROM Books WHERE Sales>=100	Searches the Books table for records where Sales is greater than or equal to 100. Returns only the Title column

2.2 Programming Fundamentals

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

File Handling Operations

- Files can be opened for reading or writing
- Append mode adds to the end of the file
- Write mode overwrites existing content in the file

`myFile := OPEN ("test.txt") FOR READING`	Opens test.txt in read mode into the myFile variable
`WHILE NOT myFile.EOF OUTPUT myFile.READLINE() END WHILE`	Uses a while loop to output each line of the file (READLINE) until the end of file (EOF) is reached.
`myFile.CLOSE()`	Closes the file

`myFile=OPEN (“logfile.txt") FOR APPEND`	Opens the logfile.txt file in append mode, meaning the existing content is preserved
`myFile.WRITELINE(“This is a log entry”)`	Writes to the end of the file
`myFile.CLOSE()`	Closes the file

`myFile=OPEN (“textfile.txt") FOR APPEND`	Opens the textfile.txt file in write mode, meaning the existing content will be overwritten
`myFile.WRITELINE(“This is a log entry”)`	Writes content to the file
`myFile.CLOSE()`	Closes the file

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

```
IF X > 50 THEN
      OUTPUT “A*”
ELSE IF X > 30 THEN
  OUTPUT “A”
ELSE
      OUTPUT “Fail”
END IF

SELECT CASE X
        CASE >100
              OUTPUT “A*”
        CASE >80
              OUTPUT “A”
        CASE >60
              OUTPUT “B”
        CASE ELSE
              OUTPUT “Fail”
END SELECT
```

Iteration

- Running through or ‘iterating’ through a set of steps several times.
- Also known as looping
- Count controlled 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 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.