Algorithms

An **algorithm** is a sequence of steps that can be followed to complete a task

- logically followed with a predictable outcome each time

A computer program is an implementation of an algorithm

- There may be more than one way to implement an algorithm

Algorithms

Algorithms can be complex.

Decomposition is the breaking down of a problem into a series of sub-problems

- Each sub-problem accomplishes an identifiable task
- Sub-problems can be broken down further into their own sub-problems

This makes dealing with the problem easier

Algorithms

Abstraction is the process of removing unnecessary detail from a problem

- This makes the problem easier to understand
- The detail might become a sub-problem which can be dealt with using decomposition or might be dealt with using a function library of some kind
- Example on next slide…

Algorithms

```
myArray <- [“orange”, “apple”, “banana”, “date”]
myFruit <- USERINPUT
myArray[4] <- myFruit
sort myArray alphabetically
FOR i <- 0 TO LEN(myArray)
    OUTPUT myArray[i]
ENDFOR
```

The line in bold is **abstracted**. You don’t need to know **how** to sort the array, just that it can be done

Algorithms

Algorithms have:

- **inputs** – data inputted, perhaps from a user
- **processes** – things that happen to data
- **outputs** – data outputted, perhaps to screen

For the algorithm on the previous slide, can you identify inputs, processes and outputs?