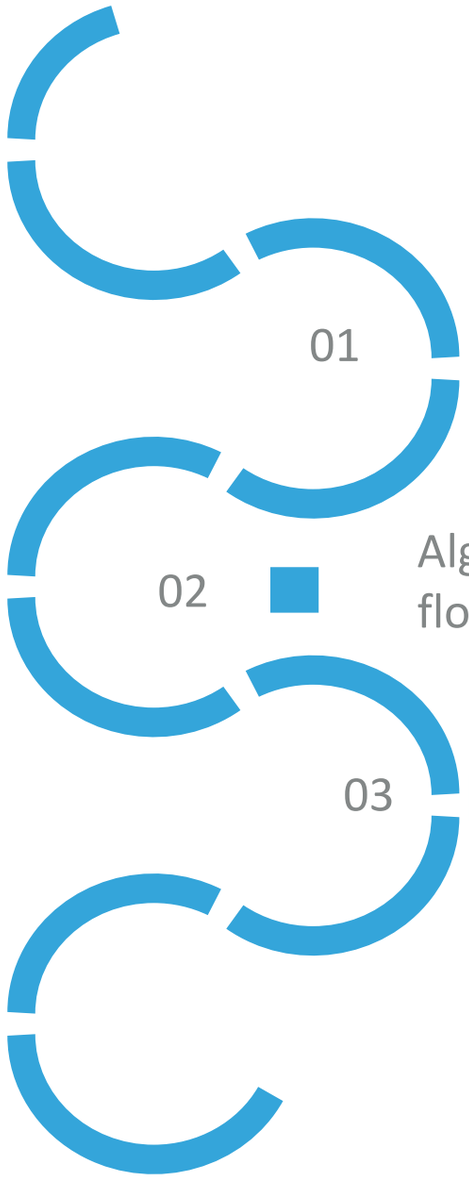
MODULE OVERVIEW

TU857
INTRODUCTION TO ALGORITHMS

CIARAN KELLY

COURSE OUTLINE



01 Data and Data types

02 Algorithms – Pseudocode & flowcharts

03 Algorithms Analysis

- REPRESENTING ALGORITHMS
- BASIC ALGORITHM CONSTRUCTS
- SEQUENCE
- SELECTION
- REPETITION
- WRITING PSEUDOCODE
- WRITING FLOWCHARTS
- TYPICAL SAMPLE ALGORITHMS

01 Labs&Quizzes 20%

Labs: Every Tuesday 2pm

02 Assignment 20%

Week 7

03 Classwork

Lecture: Every Tuesday 3-5pm

Tutorial: Every Wednesday 5pm

04 Final Exam

Week 14

In person

- Introduction to Algorithms CMPU1014

COURSE OUTLINE

01

Analysis

Computational Thinking,

Complexity -> big-O notation.

02

Techniques

How to sort data – simple sorts, complex sorts

Searching

03

Data Structures

Linked lists -> Binary Search Trees.

COURSE OUTLINE

- Basic data structures.
- Arrays 1D and 2D
- Stacks
- Queues
- Linked lists
- Trees
- Binary search trees

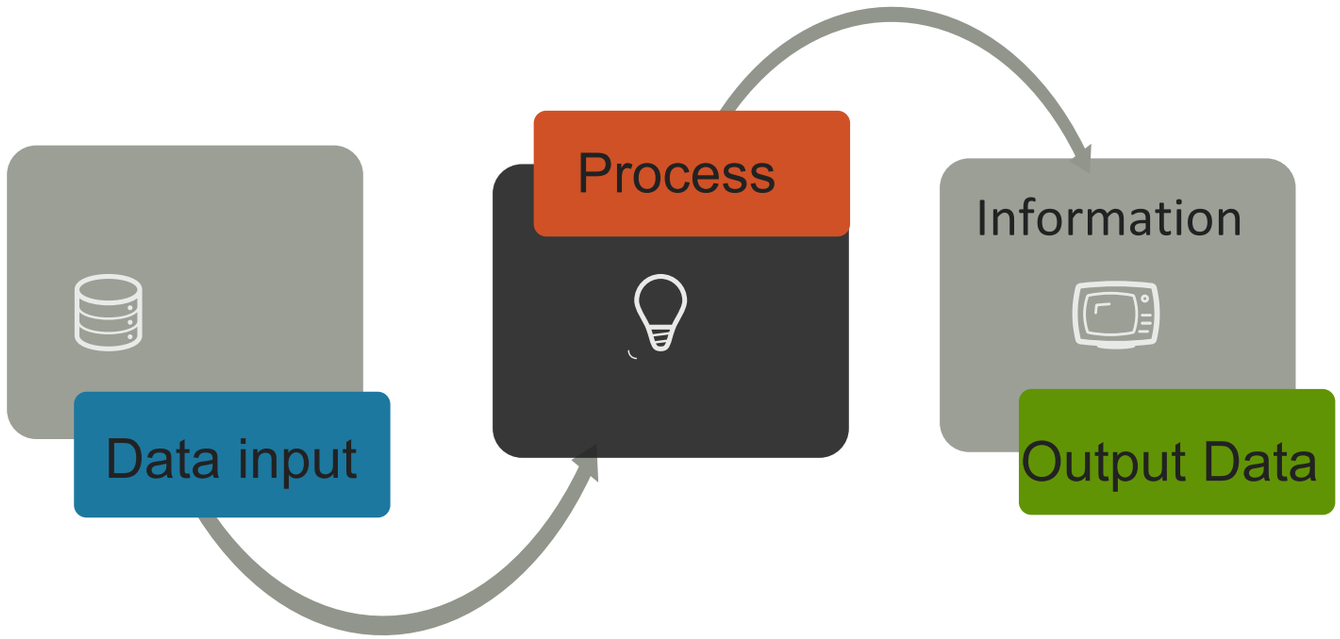
- Algorithm Analysis
- Algorithm Techniques e.g. recursion
- Computational Thinking – Decomposition, Pattern Recognition, Abstraction, Algorithms

- Doing things and solving problems – sorting, searching

Introduction

INTRODUCTION -- ALGORITHMS

- A computer (system) is a data processing machine. A simple model of a computer is shown as follows.



- Computer processes data to produce information – a result
- Input data may/may not be intelligible to humans, e.g. barcodes, electronic voltages from keyboard presses, but the result of processed information should be meaningful to humans.

INTRODUCTION -- ALGORITHMS

RESULT

Ingredients
- **unsalted butter** 175g, at room temperature
- **golden caster sugar** 175g
- **self-raising flour** 175g, sifted
- **baking powder** 1 tsp
- **vanilla extract** 1 tsp
- **eggs** 3
- **milk** 1-2 tbsp (optional)

DATA

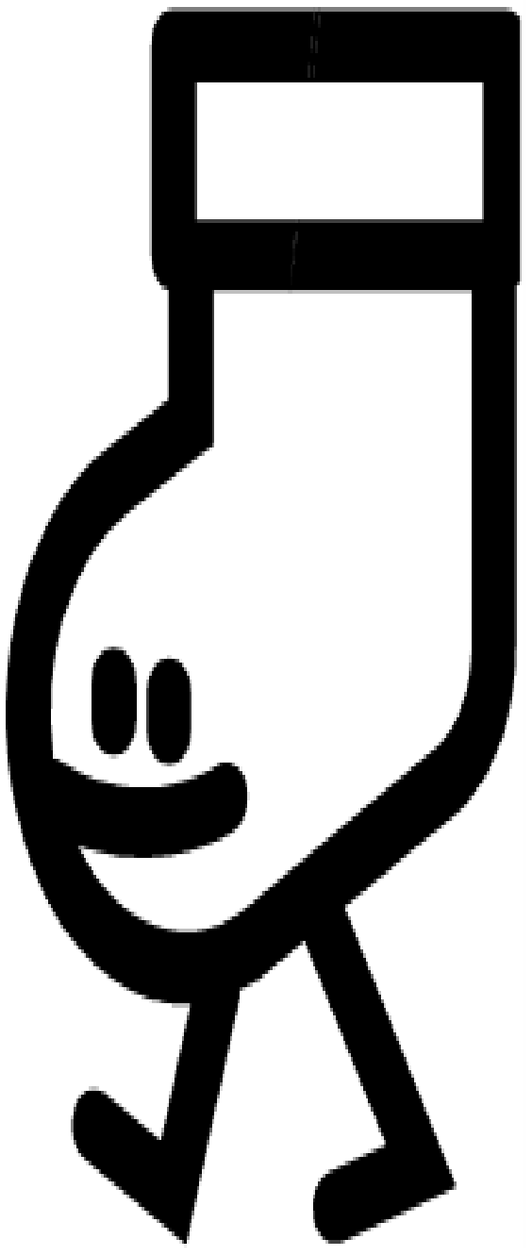
VICTORIA SPONGE FILLING
- **strawberry jam 4 tbsp, double cream 142ml, whipped**
- **icing sugar for dusting**

Method

- **STEP 1** Heat the oven to 180c/fan 160c/gas 4. Line and butter 2 x 18cm sandwich tins.
- **STEP 2** Beat all the cake ingredients together in a large bowl, add the milk if the mixture is too stiff to drop off a spoon when tapped gently.
- **STEP 3** Divide the mixture between the tins and level.
- **STEP 4** Bake side by side for 20-25 minutes until the sponges are risen, slightly shrunk away from the edge of the tin and spring back when lightly pressed.
- **STEP 5** Leave to cool for 5 minutes then turn out onto a rack and peel off the paper. Cool completely before filling.
- **STEP 6** Spread the jam onto the base of one sponge. Spread the cream on top of the jam. Sandwich the other sponge on top. Dust with icing sugar.

PROCESS

INTRODUCTION -- ALGORITHMS



Put on socks

- Rules :-
- must have 2
- must be matching?

```
1)  set socks_on_feet = 0
2)  while socks_on_feet != 2
3)      open sock drawer
4)          look for sock
5)          if you find a sock then
6)                  put on sock
7)                  socks_on_feet++
8)                  look for matching sock
9)                  if you find a matching sock then
10)                         put on matching sock
11)                         socks_on_feet++
12)                         close sock drawer
13)                 else
14)                         remove first sock from foot
15)                         socks_on_feet--
16)         else
17)                 do laundry and replenish sock drawer
```

INTRODUCTION -- ALGORITHMS

- Computers process data using computer programs to carry out specific tasks.
- These programs are built using Data Structures and Algorithms.
- **Data Structures** are the means that data can be stored and manipulated in a suitable way by computer programs. These may be simple or may be more complicated.
- **Algorithms** describe the set of instructions that must be carried out to solve a particular problem or task. These may also be simple or quite complicated depending on the problem.

INTRODUCTION -- ALGORITHMS

- **Data.** A basic problem is to understand how data is stored in a computer. This is necessary to understand, in order to create efficient and useful algorithms.
- **Representing and Storing Data.**
  - All data stored in a computer is represented in binary form. This means that only 2 distinct symbols (signals) need to be recognised. We denote these numerically by 0 and 1. We call numbers in this form base 2 numbers, or binary numbers.

INTRODUCTION -- ALGORITHMS

- Our usual number systems uses base 10, i.e. we have 10 different symbols to recognise: 0, 1, 2,…, 9.
- Basic unit of computer storage is the bit (1 **bi**nary digi**t** = 1 bit)
- 1 Byte = a set of 8 bits, stored together.
- 1 Word = set of 4 bytes or 32-bits.
- Sometimes 1 Word or a Long Word = 8 bytes or 64 bits.

INTRODUCTION -- ALGORITHMS

- Computer **memory (RAM)** is usually accessed or located using 32-bits. This is because a large amount of memory needs many bits to specify every location.
- Using 32 bits we can access or address 2^{32} memory locations. This is a large number, about 4.3 billion locations.

ALGORITHMS DEFINITION

- **Definition**. An algorithm is a finite set of instructions that specify a sequence of operations to be carried out in order to solve a specific problem or class of problems.

- An algorithm must have the following properties:

- **Finiteness**: Algorithm must complete after a finite number of instructions have been executed.

- **Uniqueness**: Each step must be clearly defined , having only one interpretation.

- **Sequence**: Each step must have a unique defined preceding and succeeding step. The first step (start step) and last step (halt step) must be clearly noted.

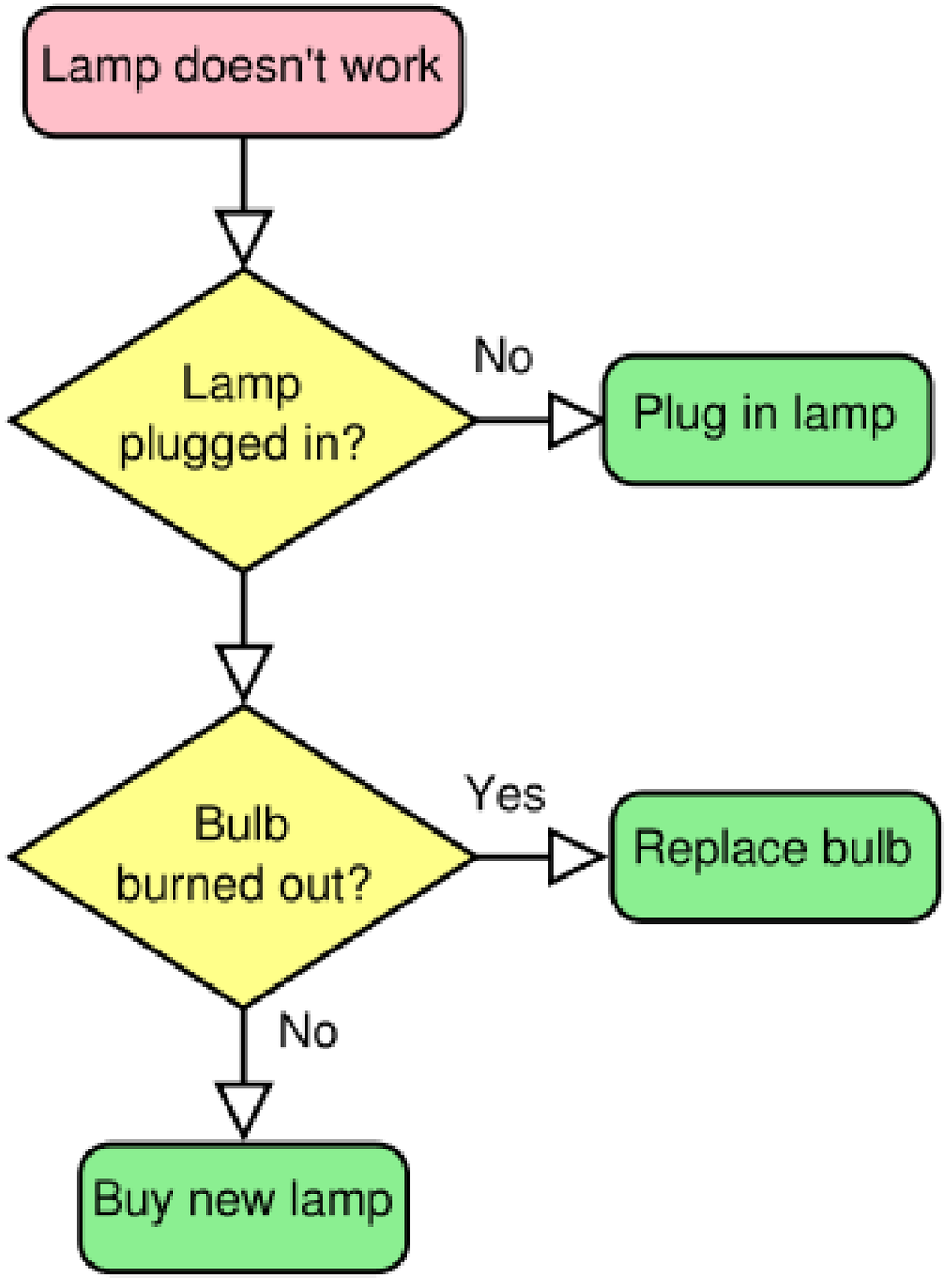
- **Feasible:** It is possible to execute each step in the algorithm

EXAMPLE OF AN ALGORITHM

DESCRIBE A POSSIBLE SOLUTION FOR A GIVEN PROBLEM:

- ☑ Identify the Input
- ☑ Identify the logical steps to solve the problem
- ☑ Identify the possible output(s)

LAMP DOESN'T WORK



ALGORITHMS

- **Exercise:**

- **Write an algorithm that finds the maximum element of 3 numbers - a,b,c.**

ALGORITHMS

- **Algorithm 1.** Finding the Maximum of 3 numbers.

```
TITLE:   FIND THE MAXIMUM OF THREE NUMBERS
INPUT :      THREE NUMBERS A, B, C
OUTPUT :       X, THE LARGEST OF A, B AND C.

1. MAX(A,B,C)          //  THE ALGORITHM NAME

2. X := A
3. IF B > X THEN         // B IS LARGER THAN X, UPDATE X
4.     X := B
5. IF C > X THEN         // C IS LARGER THAN X, UPDATE X
6.     X := C

7.   RETURN (X)
8. END MAX
```

ALGORITHMS

Input/output: There must be a specified number of input values, and one or more result values.

Feasibility: It must be possible to perform each instruction.

Another way to do the max algorithm:

1. x := a
2. if b > x, then x := b
3. if c > x, then x := c

- Idea is to inspect the numbers one by one and copy the largest value seen into x.
- At the end of algorithm, x will then be equal to the largest of the 3 numbers.
- Pseudocode: -
  - The **:=** is the **assignment operator**, i.e. copy value of a into x.
    - pi = 3.14159
    - x:= 5, pi := 3.1425
  - In **C** it is just **=** operator.
  - equality '==' vs assignment '='

TRACE YOUR ALGORITHM

- We show how the algorithm executes for some specific values of a, b and c.
- Such a simulation is called a **trace** or dry-run. Suppose that a=1, b=5, c =3.
- At line 1, set x to 1, At line 2 we set x to b=5, 5 > 1.
- At line 3, c > 5 is false, so do nothing.
- So now x = 5 , largest of (1,5,3).

```
a=1. b=5, c=3
1. x := 1
2. if b > x, then x := b
3. if c > x, then x := c

=> x is 5
```

```
Value of x
1. x = 1
2. x = 5
3. x = 5

=> x is 5
```

- Suppose that a = 6, b = 1, c = 9, then trace the algorithm.
- Note that our example algorithm has the properties set out earlier.

ALGORITHMS PROPERTIES

- The algorithm has these properties:-
  - It stops after finitely many steps giving the result.
  - Given values of the input each step of the algorithm produces a unique result.
  - There is a definite sequence of steps -*1, 2, 3*.
  - The algorithm receives input and produces output. It receives input a,b,c and produces output, max value x.
  - The steps are feasible.

ALGORITHMS

- Notation for Algorithms.
- Ordinary language is sometimes used, but many computer scientists prefer **Pseudocode,** because of its precision structure and universality.
- Pseudocode is so named because it resembles the actual code (programs) of languages such as Pascal and C. Many versions of pseudocode. Unlike actual computer languages, any type of pseudocode is acceptable, as long as its instructions are unambiguous and it resembles in form, if not in exact syntax the language used (or pseudocode here).

ALGORITHMS

- Our algorithms consist of a title, a brief description, the input and output, and the procedures containing the instructions, here a single procedure.

- The first line contains the algorithm name, and then in brackets the parameters supplied to the procedure. In line 7 we return the value of x, the largest of the numbers.

- The last line consists of the word “end” followed by the algorithm name.

ASSIGNMENTS

- Simplest statement is the assignment, giving a value to a variable (data item).
- Variable is a user-defined name that holds data item.

x := 5

pi := 3.14159

Name := “john”, etc

So assignments look like:

variable := data / variable / expression

ASSIGNMENTS

- Consider the simple example.

- length := 5.2
- width := 6.3
- area := length * width
- perimeter := 2*length + 2*width

- Here the variables are chosen to be self-explanatory.
- The variable must be on the LHS.
- You should always do this if possible.

ASSIGNMENTS

- **Assignments are used to:**
- Initialise values to variables.
- Assign constant values.
- Compute expressions, e.g. math formulas, using other variables or constants.
- **Example:**
- pi := 3.14159
- radius := 2
- Area := pi*radius*radius

TYPES OF STATEMENTS

- There are three types.
  - **Sequence.**
  - **Selection (decisions)**
  - **Iteration (repeating statements)**

- Sequence
  - Successive lines of code, no branches no loops

```
DIM x INTEGER
LET x =1
LET x = doAFunction()
PRINT x
```

SELECTION – IF

- If statements are used to make selections.
- There are **three main types**.
- Type1: If condition then statement, or

```
    If condition then      //  write a comment here
        statement(s)
   end if
```

```
If  a < b then          // if this is true (this is a comment)
    small := a
end if
```

SELECTION – IF

- Type2: If **condition is true or false** then different action.

```
If condition then      // condition is true
    statement(s)
else                   // condition is false
    statement(s)
end if
```

```
If a < b then
    smaller := a
else
    smaller := b
end if
```

SELECTION – IF

- 3.Type3: If statement with **several different conditions.**

```
If condition1 then          // condition1 is true
        statement(s)
else if condition2 then     // condition2 is true
        statement(s)
else if condition3 then     // condition3 is true
        statement(s)
….
else
        statement(s)
end if
```

- Most often we use **type 1 or 2 If statements.**

ITERATION

- **Type 1 – while loop**

while (boolean-expression)
do
  <stmt 1>
  ..
  <stmt n>
done

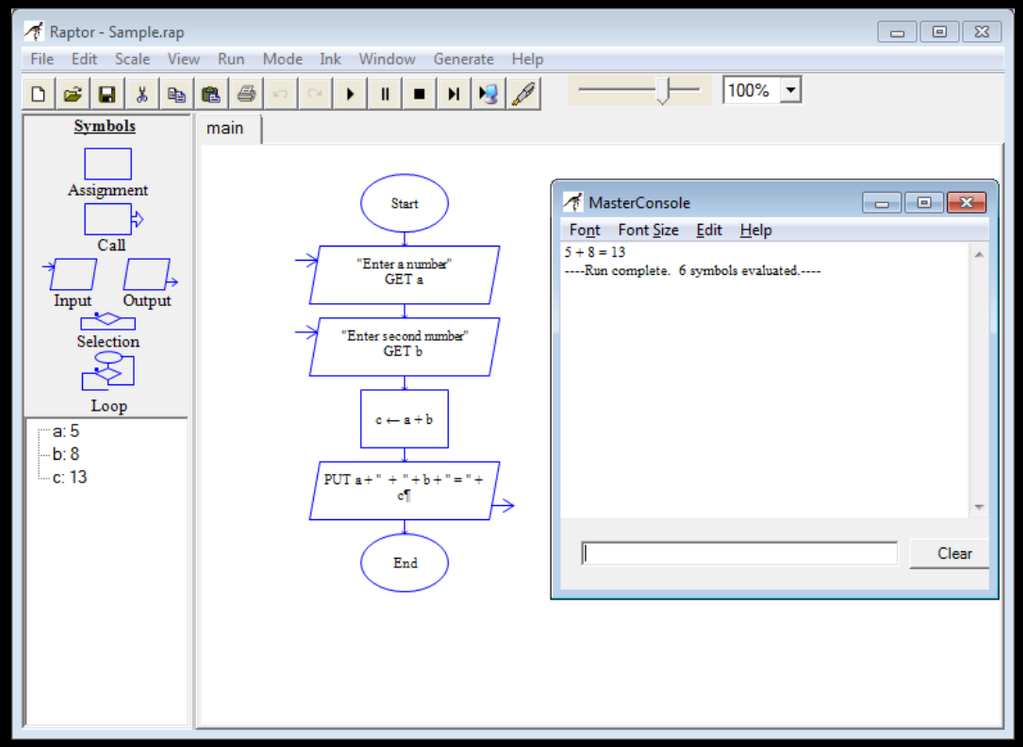
e.g. While (1) do print (“something”) done

- **Type 2 – for..next**

For indexVariable initialized to <StartValue> FROM <BeginValue> TO <EndValue>
do
  <stmt 1 using indexVariable etc>
  ..
  <stmt n>
done

RAPTOR

- RAPTOR is a flowchart-based programming environment, designed specifically to help students visualise their algorithms
- C code to implement algorithms and test their performance



http://raptor.martincarlisle.com

DEMO OF RAPTOR

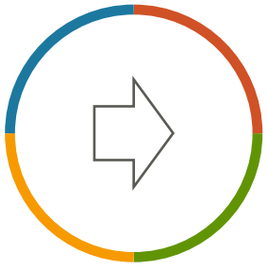
ALGORITHMS

- **Exercise:**

- *Before next week, write an algorithm that finds the second largest element of a, b and c.*

- *Implement this in Raptor – animate it and test it for a range of numbers. Can you get it to fail?*

- *Write the standard method of adding 2 positive integers, taught in primary school, as an algorithm.*



THANK YOU