Topic: 2.1.1 Algorithms

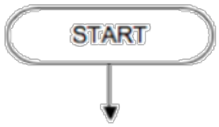
What is an Algorithm?

An algorithm is a sequence of steps, which perform a specific task. In computing, algorithms are usually represented as a program flowchart, or in pseudo-code.

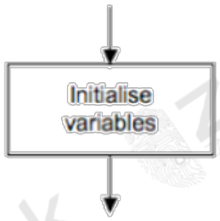
Program flowchart

A program flowchart is a pictorial representation of an algorithm. Program flowcharts use special symbols:

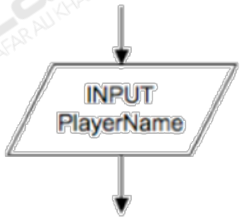
Start/stop symbol



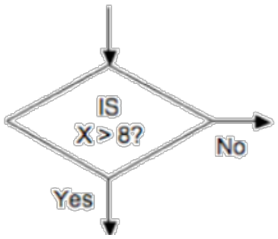
Process symbol



Input/output symbol



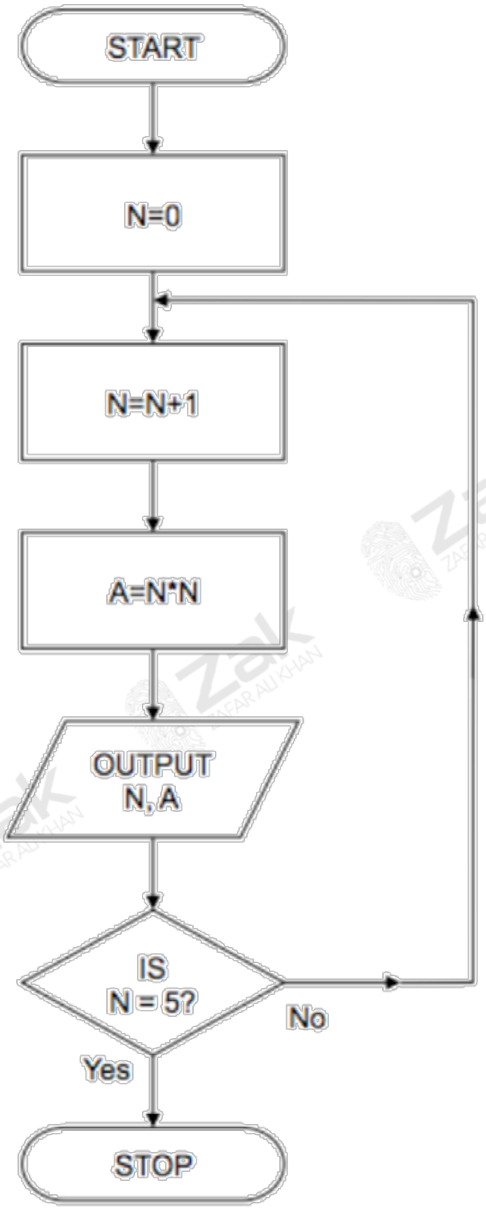
Decision symbol



Topic: 2.1.1 Algorithms

Example:

Flowchart to output the first five square numbers:



Topic: 2.1.1 Algorithms

Pseudo-code

Pseudo-code is a simplified form of programming code that uses common programming keywords, but does not use the strict syntax rules of a programming language.

An example of a pseudo-code algorithm:

```
BEGIN
      INPUT CardNumber
      REPEAT
            INPUT PIN
            IF PIN is wrong for this CardNumber THEN
                  OUTPUT “Wrong Pin”
            END IF
      UNTIL Pin is correct
      INPUT Amount
      IF there are enough funds THEN
            Dispense Cash
            Update customer’s balance
      ELSE
            OUTPUT “Sorry, insufficient funds”
      END IF
END
```

Topic: 2.1.1 Algorithms

Meaningful identifier names

Identifiers are used to give names to constants and variables. They are also used to name procedures, functions and the main program.

Naming conventions

Most of the identifier names must conform to the following rules (different programming languages may have slightly different rules):

1. they must be unique;
2. spaces must not be used;
3. they must begin with a letter of the alphabet;
4. the rest of the identifier must **NOT** contain punctuation – it may only consist of a mixture of letters and digits (A–Z, a–z and 0–9) and the underscore character '_';
5. they must not be a 'reserved' word – e.g. Print, Repeat, For, Dim, Loop, etc.

Recommended naming policies

Do not use spaces within identifier names – even with programming languages where they are permitted. Instead, use the underscore character '_' or, better yet, type names in lowercase except the first letter of each word, which should be typed in uppercase.

Examples of good identifier names:

```
FirstName            LastName            PostCode
TelephoneNumber      WeightAtBirth       TestScore
AverageHeight
```

Further clarity can be given to identifier names by including a prefix that identifies the data type.

The above identifiers would be clearer if given the following prefix data types:

```
strFirstName         strLastName         strPostCode
strTelephoneNumber   sglWeightAtBirth    intTestScore
sglAverageHeight
```

Topic: 2.1.1 Algorithms

Algorithm Basic Constructs

Assignment

An assignment is an instruction in a program that places a value into a specified variable.

Some typical assignments are:

- TheLength = 20.5
- TheUsersName$ = "Charlie"
- TheArea = TheWidth * TheLength
- TotalCost = LabelledCost + 15
- Counter = Counter + 1

Note that the last example is a common method used to increment the value of a variable. It could be read as:

"The new value of Counter is its existing value plus one"

Type Mismatch errors

A type Mismatch error occurs in a program when a variable has been declared as one data type, but it is later assigned a value that is of an incompatible data type.

The following code will produce a 'Type Mismatch' error because "Charlie" is not an integer:

```
DIM MyCounter AS Integer
MyCounter = "Charlie"

Other Type Mismatches will be produced by the following:

DIM RentalDateAs Date
MemberRentalDate = "September"

DIM ShoeSizeAs Integer
JohnsShoeSize = 10.3
```

Note that a variable that is declared as a string will never produce a type mismatch error.

Topic: 2.1.1 Algorithms

Sequence

Sequence is when the programming statements are executed one after the other, in the order in which they appear in the program.

Selection

Selection is a control structure in which there is a test to decide if certain instructions are executed.

IF-THEN-ELSE

This selection method is used if there are two possible outcomes to a test:

```
IF x < 0 THEN
      OUTPUT "Sorry, you can't have negative values"
ELSE
      a = x*x
      OUTPUT a
END
```

SELECT-CASE

This selection method is used if there are more than two possible outcomes to a test:

```
SELECT CASE KeyPress
      CASE LeftArrow
            Move one character backwards
      CASE RightArrow
            Move one character forwards
      CASE UpArrow
            Move one character up
      CASE DownArrow
            Move one character down
END SELECT
```

Topic: 2.1.1 Algorithms

Nested selection

This is where there is an IF statement within an IF statement.

The following algorithm allows a maximum of four attempts to login to a computer system:

```
INPUT Password
IF NumberOfTries< 5 THEN
      IF Password is correct THEN
            OUTPUT “Successful Login”
      ELSE
            OUTPUT “Password was incorrect”
      ENDIF
ELSE
      OUTPUT “You have made too many attempts”
ENDIF
```

Nested iteration

This is where there is a loop within a loop.

A nested iteration is needed to initialize a two-dimensional array:

```
FOR row = 0 TO 7
      FOR column = 0 TO 5
            SET MyArray (row, column) = 0
      NEXT column
NEXT row
```

Iteration

Iteration is a control structure in which a group of statements is executed repeatedly – either a set number of times, or until a specific condition is True.

FOR-NEXT

This is an unconditional loop in which the number of repetitions is set at the beginning.

```
FOR X = 1 TO 5
      Answer = X*3
      OUTPUT X, Answer
NEXT
```

Topic: 2.1.1 Algorithms

WHILE-ENDWHILE

This is a conditional loop, which has a test at the start and repeats until the condition is false:

```
X = 0
WHILE X < 6 DO
      X = X + 1
      Answer = X*3
      OUTPUT X, Answer
ENDWHILE
```

REPEAT-UNTIL

This is a conditional loop, which has a test at the end and repeats until the condition is true:

```
X = 0
      REPEAT
      X = X + 1
      Answer = X*3
      OUTPUT X, Answer
UNTIL X > 4
```

Structured English	Pseudo Code	Executable Code

BEGIN
READ name
IF name EQUAL "Harry" THEN
WRITE "Why don't you marry Pippa?"
ELSE
WRITE "Are you Royal enough?"
END IF
END
	BEGIN
	INPUT name
	IF name == "Harry" THEN
	OUTPUT "Why don't you marry Pippa?"
	ELSE
	OUTPUT "Are you Royal enough?"
	END IF
	END
		dim name as string
		name = console.readline()
		if name = "Harry" then
		console.writeline("Why don't you marry Pippa?")
		else
		console.writeline("Are you Royal enough?")
		End if

Topic: 2.1.1 Algorithms

Comparison of the different iterations:

FOR-NEXT	WHILE-WEND	REPEAT-UNTIL
An unconditional loop – it will repeat itself a set number of times.	A conditional loop – it will repeat until a variable has reached a stated value.	A conditional loop – it will repeat until a variable has reached a stated value.
	The condition is tested at the start of the loop.	The condition is tested at the end of the loop.
Iteration statements will always execute at least once.	Iteration statements may not actually be executed.	Iteration statements will always execute at least once.

`For x= 1 To 10`
`  a=x*5`
`  Output x, a`
`Next x`
	`x=1`
	`While x < 11`
	`  a=x*5`
	`  Output x, a`
	`  x=x+1`
	`End While`
		`x=1`
		`Repeat`
		`  a=x*5`
		`  Output x, a`
		`  x=x+1`
		`Until x > 10`

Top down/modular design

Top-down design is when a problem is split into smaller sub-problems, which themselves are split into even smaller sub-problems until each is just one element of the final program.

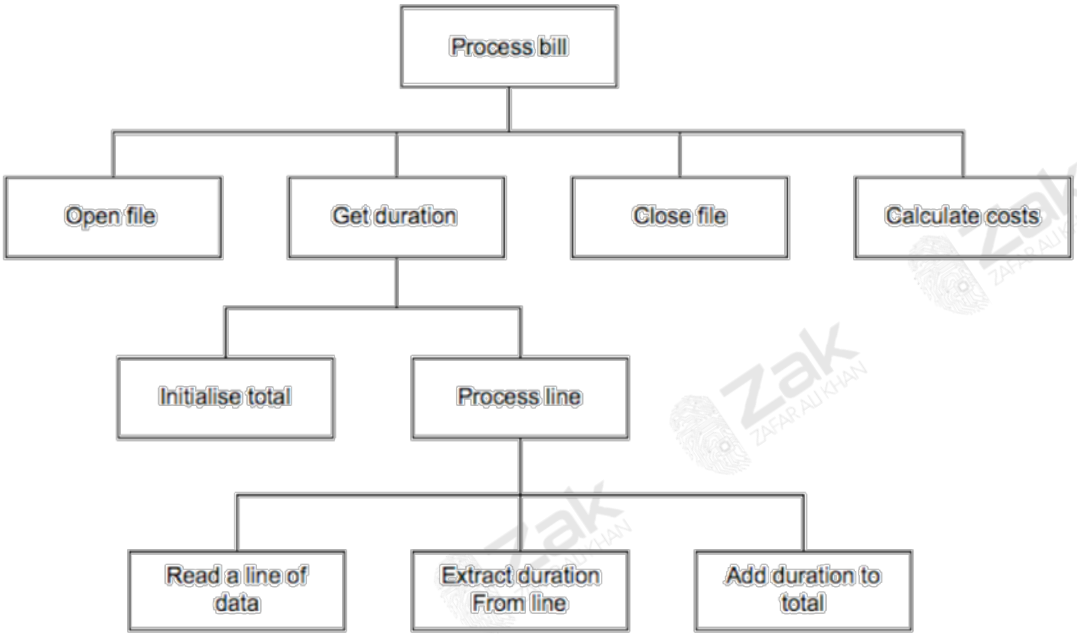
Benefits and drawbacks of modular programs:

Benefits	Drawbacks
Smaller problems are easier to understand	Modules must be linked and additional testing must be carried out to make sure that the links work correctly
Smaller problems are easier to test and debug	Programmers must ensure that cross-referencing is done
Development can be shared between a team of programmers – each person programming different modules according to their individual strengths.	Interfaces between modules must be planned
Code from previously developed modules can be re-used	

Topic: 2.1.1 Algorithms

Structure diagrams

A structure diagram is a pictorial representation of a modular system.



Stepwise refinement

Stepwise refinement is the process of developing a modular design by splitting a problem into smaller sub-tasks, which themselves are repeatedly split into even smaller sub-tasks until each is just one element of the final program.

Topic: 2.1.1 Algorithms

Subroutine

A subroutine is a self-contained section of program code that performs a specific task, as part of the main program.

Procedure

A procedure is a subroutine that performs a specific task without returning a value to the part of the program from which it was called.

Function

A function is a subroutine that performs a specific task and returns a value to the part of the program from which it was called.

Note that a function is 'called' by writing it on the right hand side of an assignment statement.

Parameter

A parameter is a value that is 'received' in a subroutine (procedure or function).

The subroutine uses the value of the parameter within its execution. The action of the subroutine will be different depending upon the parameters that it is passed.

Parameters are placed in parenthesis after the subroutine name. For example:

Square(5) 'passes the parameter 5 – returns 25

Square(8) 'passes the parameter 8 – returns 64

Square(x) 'passes the value of the variable x

Topic: 2.1.1 Algorithms

Subroutine/sub-program

A subroutine is a self-contained section of program code which performs a specific task and is referenced by a name.

A subroutine resembles a standard program in that it will contain its own local variables, data types, labels and constant declarations.

There are two types of subroutine. These are procedures and functions.

- Procedures are subroutines that input, output or manipulate data in some way.
- Functions are subroutines that return a value to the main program.

A subroutine is executed whenever its name is encountered in the executable part of the main program. The execution of a subroutine by referencing its name in the main program is termed 'calling' the subroutine.

The benefits of using procedures and functions are that:

- The same lines of code are re-used whenever they are needed – they do not have to be repeated in different sections of the program.
- A procedure or function can be tested/improved/rewritten independently of other procedures or functions.
- It is easy to share procedures and functions with other programs – they can be incorporated into library files which are then 'linked' to the main program.
- A programmer can create their own routines that can be called in the same way as any built-in command.