ZNOTES.ORG

UPDATED TO 2017-19 SYLLABUS

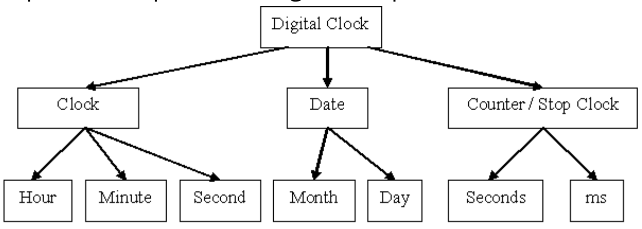
CAIE IGCSE COMPUTER SCIENCE (0478)

SUMMARIZED NOTES ON THE PRACTICAL SYLLABUS

1. Algorithm Design & Problem-Solving

1.1. Problem-solving & Design

- Every computer system is made up of sub-systems, which are in turn made up of further sub-systems.
- **Top-down Design** – The breaking down of a computer system into sub-systems, then breaking each sub-system into smaller sub-systems, until each one only performs a single action. A structure diagram diagrammatically represents top-down design. Example below.

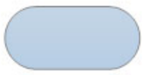
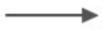
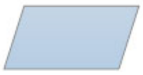
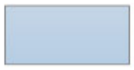
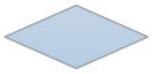


- **Test data** – All the items of data required to work through a solution. It is inputted into the program and compared with the expected results. Examples are for a school grade
  - Normal – 28; 64; 98 - Accept
  - Erroneous/Abnormal – eleven; -12; 158 - Reject
  - Extreme – 0; 100 – Accept
  - Boundary – 0; -1 – Accept; Reject
- **Validation** – Automated checking by a program that data is reasonable before it is accepted as an input.
  - Range – Accepts numbers within a specified range
  - Length – Accepts data with an exact number of characters OR has a reasonable amount of characters
  - Type – Accepts data with a certain data type
  - Character – Accepts data without invalid characters
  - Format – Accepts data that conforms to a specified patter/format
  - Presence – Requires data to be inputted
- **Verification** – Checking that data has been accurately copied onto the computer or transferred from one part of a computer system to another.
  - Double entry – Data is entered twice and compared
  - Visual/Screen – Manual check compared by the user
- **Sub-routine** - Block of code that can be called and accessed by a main program.
- Functions are sub-routines that return a single value
- **Trace Tables:** A technique used to test algorithms, in order to make sure that no logical errors occur whilst the algorithm is being processed.

		x	y	z	x > 0
1	x=5	5			
2	y=1		1		
3	z=0			0	
4	while x>0:				T
5	x=x-1	4			
6	y=y+1		2		
7	z=(x+y)*2			12	
4	while x>0:				T
5	x=x-1	3			
6	y=y+1		3		
7	z=(x+y)*2			12	
4	while x>0:				T
5	x=x-1	2			
6	y=y+1		4		
7	z=(x+y)*2			12	
4	while x>0:				T
5	x=x-1	1			
6	y=y+1		5		
7	z=(x+y)*2			12	
4	while x>0:				T
5	x=x-1	0			
6	y=y+1		6		
7	z=(x+y)*2			12	
4	while x>0:				F
		0	6	12	

1.2. Pseudocode & Flowcharts

- **Pseudocode -** Verbal representation of an algorithm (a process or set of steps) and flowcharts are a diagrammatic representation.
- **Flowcharts**

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

- **Input & Output (READ & PRINT) –** Used to receive and display data to the user respectively

```
OUTPUT "ENTER NAME"
INPUT NAME
OUTPUT "HELLO", NAME
(ALTERNATIVELY)
PRINT "ENTER NAME"
READ NAME
PRINT "HELLO", NAME
```

- **Assignment -** Each variable is assigned using a left arrow.

```
[VARIABLE] ← [VALUE]
GRADE ← 98
```

- **Conditional Statements:**
  - IF…THEN…ELSE…ENDIF – 1 condition

```
IF [CONDITION] THEN
    [CONSEQUENCE]
ELSE
    [CONSEQUENCE]
ENDIF

IF GRADE > 100 THEN
    OUTPUT "INVALID"
ELSE
    OUTPUT "VALID"
ENDIF
```

- CASE…OF…OTHERWISE…ENDCASE – Multiple conditions and corresponding consequences

```
CASE OF [VARIABLE]
    OPTION: [CONSEQUENCE]
OTHERWISE: [CONSEQUENCE]
ENDCASE

CASE OF GRADE
    GRADE>80: OUTPUT "A"
    GRADE>70: OUTPUT "B"
    GRADE>60: OUTPUT "C"
OTHERWISE: OUTPUT "FAIL"
ENDCASE
```

- **Loop Structures:**
  - FOR…TO…NEXT- Will run for a determined/known amount of times

```
FOR [VARIABLE] ← [VALUE] TO [VALUE]
    [CODE]
NEXT
```

  - REPEAT… UNTIL – Will run at least once till condition is satisfied; Verification is done after running code

```
REPEAT
    [CODE]
UNTIL [CONDITION]
```

  - WHILE…DO…ENDWHILE – May not ever run; Verification is done before running code

```
WHILE [CONDITION] DO
    [CODE]
ENDWHILE
```

2. Programming

2.1. Programming Concepts

- Declaration & Usage of Variables & Constants
  - Variable – Store of data which changes during execution of the program (due to user input)
  - Constant – Store of data that remains the same during the execution of the program
- Basic Data Types
  - Integer – Whole Number e.g. 2; 8; 100
  - Real – Decimal Number e.g. 7.00; 5.64
  - Char – Single Character e.g. a; Y
  - String – Multiple Characters (Text) e.g. ZNotes; COOL
  - Boolean – Only 2 Values e.g. True/False; Yes/No; 0/1

```
DECLARE [VAR/CONST] AS [DATA TYPE]
← [VALUE]
```

- IMPORTANT CONCEPTS
  - Sequence – Statements are executed in order. E.g. Variables must first be declared, and then used.
  - Selection – Allows data items to be picked according to given criteria. E.g. Finding the highest/smallest value
  - Repetition – Causes statements to be repeated (loops)
  - Totalling – Used with repetition, to keep the total updated. E.g.

```
BillTotal ← BillTotal + ProductCost
```

- Counting – Used with repetition to increment the counter by 1, each time the loop is repeated. E.g.

```
NumItems ← NumItems + 1
```

2.2. Data Structures; Arrays

- Declaration

```
DECLARE [NAME][1:n] AS [DATA TYPE]

DECLARE GRADE [1:18] AS REAL
```

- Use of FOR Loop to Read & Write

```
DECLARE GRADE [1:18] AS INTEGER
FOR I ← 1 To 18
    OUTPUT "GRADE OF STUDENT", I
    INPUT/OUTPUT GRADE [I]
NEXT
```

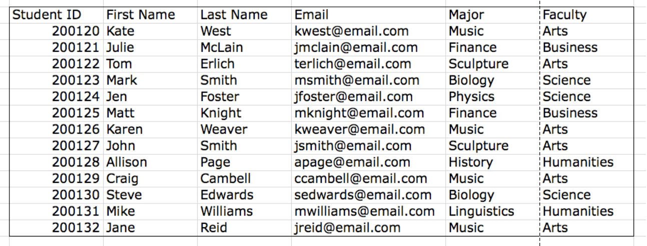
3. Databases

3.1. Data types

- The data type names are different in Access:
  - Real – Number
  - String – Text
  - Boolean – Yes/No

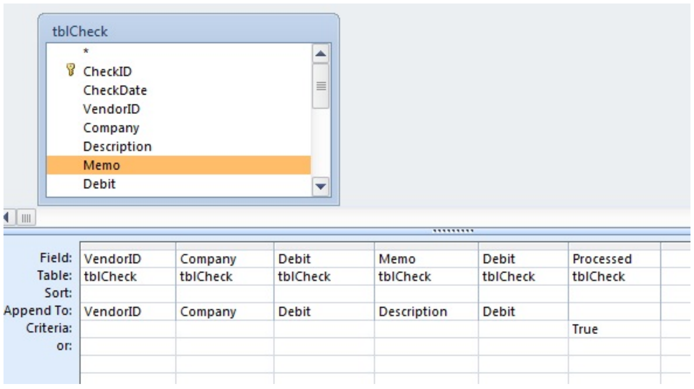
3.2. Primary Key

- It is a field that uniquely identifies each record. E.g. Student code will be the primary key in a school database.



Student ID	First Name	Last Name	Email	Major	Faculty
200120	Kate	West	kwest@email.com	Music	Arts
200121	Julie	McLain	jmclain@email.com	Finance	Business
200122	Tom	Erlich	terlich@email.com	Sculpture	Arts
200123	Mark	Smith	msmith@email.com	Biology	Science
200124	Jen	Foster	jfoster@email.com	Physics	Science
200125	Matt	Knight	mknight@email.com	Finance	Business
200126	Karen	Weaver	kweaver@email.com	Music	Arts
200127	John	Smith	jsmith@email.com	Sculpture	Arts
200128	Allison	Page	apage@email.com	History	Humanities
200129	Craig	Cambell	ccambell@email.com	Music	Arts
200130	Steve	Edwards	sedwards@email.com	Biology	Science
200131	Mike	Williams	mwilliams@email.com	Linguistics	Humanities
200132	Jane	Reid	jreid@email.com	Music	Arts

3.3. Query-By-Example (QBE)



Field:	VendorID	Company	Debit	Memo	Debit	Processed
Table:	tblCheck	tblCheck	tblCheck	tblCheck	tblCheck	tblCheck
Sort:						
Append To:	VendorID	Company	Debit	Description	Debit	
Criteria:						True
or:						

- Field: Field Name
- Table: Table Name
- Sort: Ascending (A-Z) or Descending (Z-A)
- Show: Checked (Present) or Empty (Absent)
- **Criteria**:

TEXT		
Criteria Name	Written As	Function
Contains	Like ("*x*")	Values that contain x
Does Not Contain	Not like ("*x*")	Values that do not contain x
Begins With	Like ("x*")	Values beginning with x
Ends With	Like ("*x")	Values ending with x
Comes After	>= "x"	Values that come before x in alphabetical order
Comes Before	<= "x"	Values that come after x in alphabetical order

NUMBERS		
Criteria Name	Written As	Function
Between	Between "x" and "y"	Values in the range between x and y
Less Than	<x	Values smaller than x
Less Than or Equal To	<=x	Values smaller than or equal to x
Greater Than	>x	Values larger than x
Greater Than or Equal To	>=x	Values larger than or equal to x

DATES		
Criteria Name	Written As	Function
Between	Between "#mm/dd/yyyy#" and "#mm/dd/yyyy#"	Dates between the specified dates
Before	< "#mm/dd/yyyy#"	Dates before a certain date
After	> "#mm/dd/yyyy#"	Dates after a certain date
Today	=Date()	Records containing today's date
x Days Before Today	<=Date()-x	Records containing dates x or more days in the past

CAIE IGCSE
Computer Science (0478)