THE Ebbsfleet ACADEMY
Aspire, Excel

**Year:** 10
**Topic:** Search & Sort Algorithms, Boolean logic

**Core Texts**
AQA Computer Science, Alison Page, 2013

AQA Computer Science, Robson & Heathcote, 2016

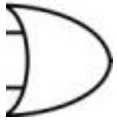
| | |
|---|---|
| **Linear search** | The most basic search algorithm. A linear search moves sequentially (one by one) through the data looking for a matching value. Does not have to be in order. |
| **Binary search** | Can only be performed on an ordered list. Binary search looks for the centre value and disregards anything above or below what we are trying to find. |
| **Bubble sort** | The simplest sorting algorithm, works by iteration (repetition). The array is sorted from the first element to the last comparing each pair of element and switching their positions if necessary |
| **Merge sort** | An example of a divide and conquer algorithm, meaning that the problem of sorting an unsorted list of items is solved by dividing the problem into sub problems, solving the sub problems and then merging the results to produce the sorted list. |

**AND**

| IN | IN | OUT |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

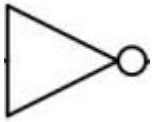
**OR**

| IN | IN | OUT |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |



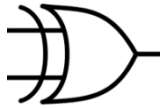
**NOT**

| IN | OUT |
|---|---|
| 1 | 0 |
| 0 | 1 |



**XOR**

| IN | IN | OUT |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |
| 1 | 1 | 0 |



**Search Algorithms compared**

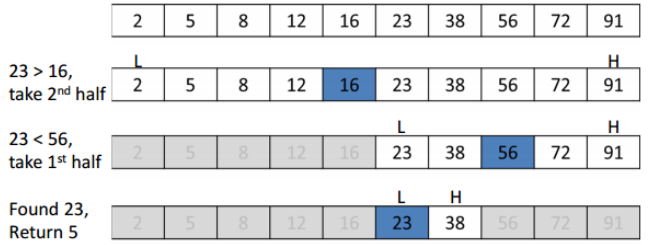
- Binary search requires the data to be sorted before it can function.
- Binary search requires random access to the data, linear search only requires sequential access
- Binary search is usually more efficient than a linear search.

**Sort Algorithms compared**

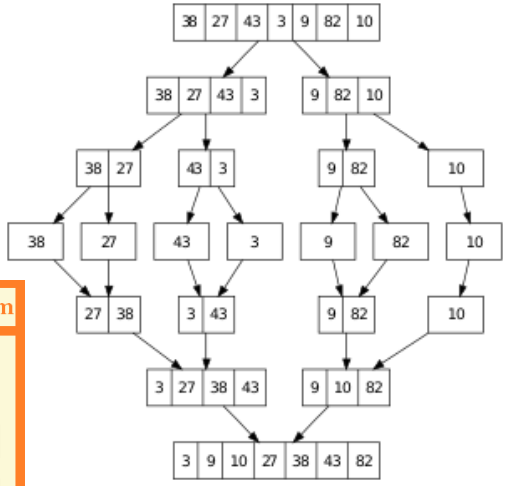
- Merge sort is well suited to sorting large amounts of data that do not fit into main memory.
- Bubble sort is an in-memory sort algorithm, all the data must be stored in main memory otherwise the bubble sort cannot happen.
- Bubble sort is usually slower to complete than merge sort as it must go through many steps
- Bubble sort is not efficient in terms of time but is quite good in terms of memory as the data is sorted within the list.
- Merge sort is more time efficient but generally uses more memory as copies of the lists are created and then split.

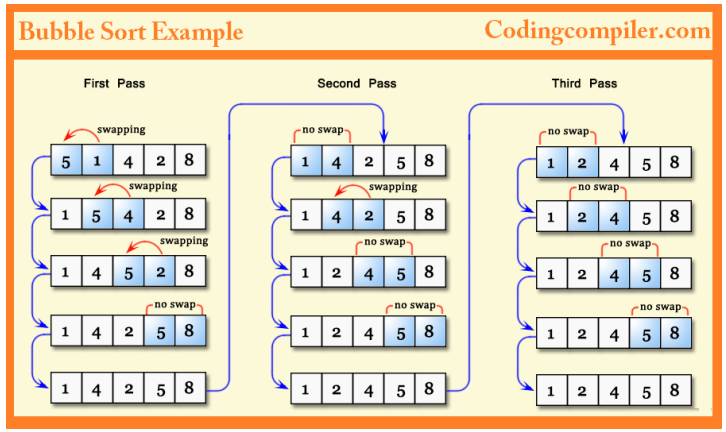
**Binary Search**

If searching for 23 in the 10-element array:




**Merge Sort**



THE Ebbsfleet ACADEMY
Aspire, Excel

**Year:** 10
**Topic:** Algorithms

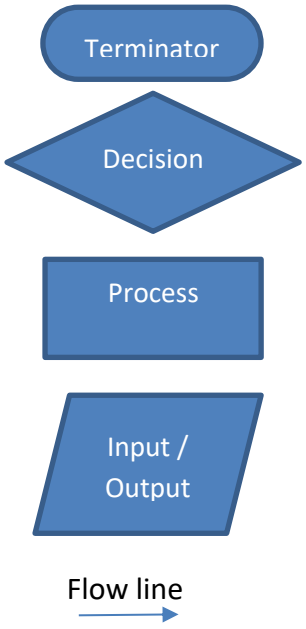
**Indefinite Iteration**
- While loop
- Repeat Until loop

**Definite Iteration**
- For loop

**Selection**
- IF statements

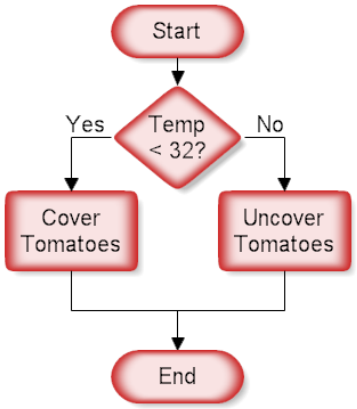
**Key Words:**

| | | | | | |
|---|---|---|---|---|---|
| **Decomposition** | Breaking down a problem into smaller steps or stages, this makes it easier to follow and solve. | **Sequence** | A sequence is a set of steps taken in order, one thing following another | **Flowchart** | A graphical way of writing an algorithm; these contain terminators, processes and decisions. |
| **Abstraction** | Involves removing unnecessary details from a problem in order to solve it, only pick out what is relevant and ignore the rest. | **Pseudocode** | Consists of natural language statements that precisely describe the steps required when constructing an algorithm | **Iteration** | Sections of code that need to be repeated a certain number of times until a certain condition is true or false. |
| **Algorithm** | A series of steps that can be followed to complete a task. | **Module** | A section of code that forms a part of an overall program. | **Arrays** | A type of list that allows you to store a group of data items of the same data type instead of creating many different variables |




Pseudocode:

| | | | |
|---|---|---|---|
| Assign a value to a variable | Variable name ← USERINPUT | Repeat Until loop | Count ← 1<br>REPEAT<br>  OUTPUT count * 3<br>  count ← count + 1<br>UNTIL count > 10 |
| Output a message to the user | OUTPUT "……." | | |
| Using If, Then, Else | IF x = 10 THEN<br>  OUTPUT "x does equal 10"<br>ELSE<br>  OUTPUT "x does not equal 10)<br>END IF | While loop | x = USERINPUT<br>WHILE x ≠ "End"<br>  OUTPUT x<br>  x = USERINPUT<br>ENDWHILE |
| For Loop | FOR count ← 1 to 10<br>  OUTPUT count * 3<br>END FOR | | |



THE Ebbsfleet ACADEMY
Aspire, Excel

## Week 1

Develop an algorithm using pseudocode or a flowchart that asks the user to create a new password.

The algorithm should:

- get the user to enter a password
- get the user to re-enter the password
- repeat the two bullet points above until both entered passwords are identical
- output "`password created`" when they are identical.

**[5 mark**

## Week 2

Develop an algorithm using either pseudo-code or a flowchart that allows a taxi company to calculate how much a taxi fare should be.

The algorithm should:

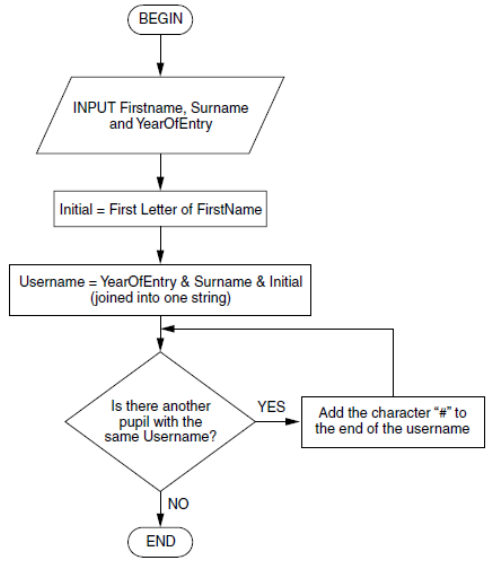
- prompt the user to enter the journey distance in kilometres
  - the distance entered must be greater than zero
  - the user should be made to re-enter the distance until the distance entered is valid
- prompt the user to enter the number of passengers (no validation is required)
- calculate the taxi fare by
  - charging £2 for every passenger regardless of the distance
  - charging a further £1.50 for every kilometre regardless of how many passengers there are
- output the final taxi fare.

**[8 marks]**

## Week 3

A school uses a computer program to give every new pupil a username for logging onto computers.

The algorithm used to choose the username is shown below.




**(a)** Mark Johnson joins the school in 2012. No other pupil called Johnson joins the school in the same year.

State the username which Mark will be given and explain how you obtained your answer from the flow diagram.

Username .....

Explanation .....

.....

.....

..... **[3]**

**(b)** A pupil has the username 2010alim###.

State **four** facts that we can work out from this username.

## Week 4

**Figure 5** shows the start of an algorithm.

**Figure 5**

```
OUTPUT 'enter the 24 hour number (0-23)'
hour ← USERINPUT
```

The algorithm in **Figure 5** asks the user to enter a number between 0 and 23 that represents an hour using the 24 hour clock. The input is stored in a variable called `hour`.

Extend the algorithm in **Figure 5**, using either pseudo-code or a flowchart, so that it outputs the equivalent time using the 12 hour clock, ie a number between 1 and 12, followed by either `am` or `pm`.

For example:

- If the user enters `0`, the program outputs `12` followed by `am`.
- If the user enters `4`, the program outputs `4` followed by `am`.
- If the user enters `12`, the program outputs `12` followed by `pm`.
- If the user enters `15`, the program outputs `3` followed by `pm`.

You can assume that the variable `hour` is an integer and that the value that the user inputs will be between 0 and 23.

**[7 marks]**

## Week 5

Develop an algorithm using pseudocode or a flowchart that calculates an estimate of the braking distance in metres for a new model of go-kart that is travelling between 10 and 50 kilometres per hour (kph).

Your algorithm should be based on the following method:

- the user should keep being asked to enter a speed for the go-kart until they enter a speed that is between 10 and 50 (both 10 and 50 are valid speeds)
- the braking distance in metres is calculated by dividing the speed by 5
- the user should be asked if the ground is wet (expect the user to enter 'yes' if it is)
- the braking distance should be multiplied by 1.5 when the ground is wet
- finally, your algorithm should output the calculated braking distance.

**[9 marks]**

## Week 6

Write an algorithm (using either **pseudocode** or a **flowchart**) that calculates the amount of fuel a train will need to complete a journey. The algorithm must:

- ask the user how many kilometres the journey will be
- only continue if the user enters a value greater than zero
- set the amount of fuel to a number 100 times greater than the number of kilometres
- not allow the amount of fuel to be less than 1500
- finally, display the amount of fuel needed.

**[7 marks]**

THE Ebbsfleet ACADEMY
Aspire, Excel

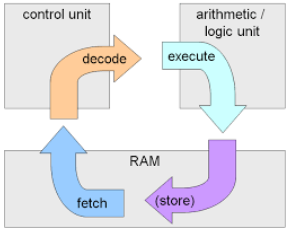
**Year:** 10
**Topic:**
CPU, Fetch Decode, Execute Cycle, Von Neumann Architecture

**Core Texts**
AQA Computer Science, Alison Page, 2013

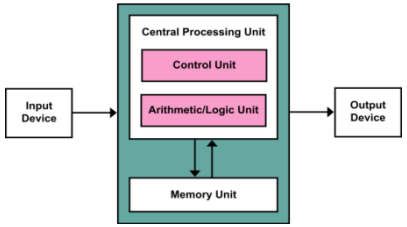
AQA Computer Science, Robson & Heathcote, 2016

**Key Words:**

| | | | | | |
|---|---|---|---|---|---|
| **Arithmetic Logic unit (ALU)** | Carries out the following 3 operations: Logical operations, shift operations, arithmetic operations. | **Fetch, Decode, Execute Cycle (FDE)** | The cycle of loading, processing and executing one instruction | **Clock frequency** | Number of clock cycles which occur each second. |
| **Control unit** | Coordinates the activities taking place inside the CPU. | **Cache Memory** | Is a middle man between RAM and the CPU. Stores frequently used instructions | **Bus** | The CPU has internal connections which pass data between the components of the CPU. |
| **Clock** | Controls processor timing, switching between 0 and 1 at rates exceeding several million times a second. It synchronises all CPU operations | **Processor Cores** | The amount of cores that can process an instruction at any given moment | **Registers** | Fast memory locations which are involved in the FDE. |




**Fetch Decode, Execute cycle (FDE)**




**Von Neumann Architecture**
Developed the idea in the 1940's of the stored program computer

ALU
**Logical operations:** These include AND, OR, NOT
**Shift operations:** The bits in a computer word can be shifted left or right by a certain number of places
**Arithmetic operations:** These include addition, subtraction, multiplication and division

Control Unit
Controls the execution of instructions in the correct sequence
Decodes instructions
Regulates and controls processor timing using regular pulses from the system clock
Sends and receives control signals to and from other devices within the computer.

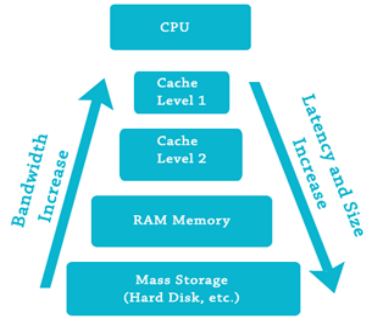
Fetch, Decode, Execute (FDE)
The address of the next instruction to be executed is fetched from the register
The register is incremented so it points to the next instruction to be fetched
The instruction is fetched and placed into a special register ready to be decoded
The Control Unit decodes the instruction to see what has to be done next.
The instruction is executed by the ALU, the result is stored back into main memory

Cache Memory
The cache makes any data frequently used by the CPU available much more quickly. Because the processor has to access main memory less often, it can work faster, so the CPU performance increases. If it is not located in cache then it has to be fetched from main memory.

A typical PC may have 8GB of RAM (Main memory) but only 2MB of the faster more expensive cache memory.

The more cache memory a computer has, the more data and instructions can be held in cache and made available very quickly.

- Level 1 cache is extremely fast but small (between 2 -64KB)
- Level 2 cache is fairly fast and medium sized (256KB – 2MB)
- Some CPU's have level 3 cache.



THE Ebbsfleet ACADEMY
Aspire, Excel

**Year:** 10
**Topic:** High & Low Level Programming Languages, ASCII, UNICODE and Operating Systems

**Core Texts**
AQA Computer Science, Alison Page, 2013

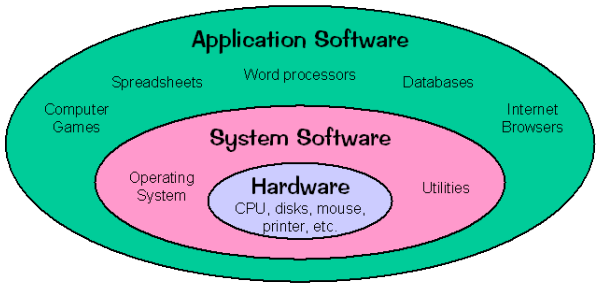
AQA Computer Science, Robson & Heathcote, 2016

**Key Words:**

| | | | | | |
|---|---|---|---|---|---|
| **Data** | Are the facts or details from which information is derived, data itself has no meaning g, but becomes information when it is interpreted | **Compiler** | A computer program that translates VB.net and similar high-level programming languages into machine code | **Low-level Programming language** | A programming language that provides little or no abstraction from a computers instruction set |
| **Information** | When data is processed, organised, structured or presented in a given context which makes it useful, it is called information | **Interpreter** | A program that executes a source program by reading it one line at a time and doing the specified operations immediately | **UNICODE** | Can use either 8, 16 or 32 bits to store each character. Only about 10% of capacity has been used so far. Used for different languages and symbols |
| **Machine Code** | The languages made up of binary coded instructions that is used directly by the computer | **High-level programming language** | A programming language designed to allow people to write programs without having to understand the inner workings of a computer | **ASCII** | American Standard Code for Information Interchange ($2^7$) values = 128 possible values |
| **Assembly Language** | At the level of telling the processor what to do | **Extended ASCII** | ($2^8$) values = 256 possible values | **OS** | A program that allows application software to communicate with hardware |

The operating system is a group of programs that manages the computers resources: Processor(s), memory, input/output devices, applications, security

Fig1. Link between software and hardware




| | How the OS controls these areas |
|---|---|
| Processor(s) | Keeps track of processes, organises processing time and manages the CPU |
| Memory | Totally responsible for memory management, keeps track of memory use and decides which process will get memory, when and how much |
| Input/output devices | Manages all devices including peripherals using drivers |
| Applications | Loads programs into memory, allows hardware to communicate |
| Security | Controls access to programs, processes and users to the computer resources. |

THE Ebbsfleet ACADEMY
Aspire, Excel

**Year:** 10
**Topic:** Programming Concepts

**Key Words:**

| | | | | | |
|---|---|---|---|---|---|
| **Global Variable** | A variable with global scope meaning that it is visible throughout the program. | **Real** | A number with a fractional part such as 34.456, -9.2, 4.10. | **String** | Zero or more characters. A string can be null (empty) just one character or many characters. |
| **Local Variable** | A variable with local scope meaning that it is only visible within the block in which it is declared. | **Boolean** | A binary variable that can have one of two possible values, 0 (false) or 1 (true). | **Concatenation** | Joining together, this can be multiple strings or variables. |
| **Testing: Erroneous** | The program is provided with data that is outside its limits of performance. These tests try to break the application and to investigate if things occur when they shouldn't and vice versa. | **Trace Table** | A technique used to test algorithms; they are used to make sure that no logic errors occur whilst the algorithm is being processed. | **Subroutine** | Is a named, self-contained section of code that performs a specific task, It may return one or more values but doesn't have to. |
| **=** | Equal to | ***** | Multiplication | **+** | Addition |
| **≠** | Not equal to | **≥** | Greater than or equal to | **/** | Division |
| **-** | Subtraction | **≤** | Less than or equal to | **Integer** | A whole number, such as 3**,** 45 |
| **Testing: Typical** | Using data that the program would accept and process under normal conditions | **Testing: Boundary** | The program is provided with data that is within the operating range but at its limits of performance | **Data Type** | Defines the type of data that is allowed to be stored in a variable, constant or array. |
| **Sequence** | A sequence is a set of steps taken in order, one thing following another | **Module** | A section of code that forms part of an overall program. | **Logical Operators** | AND, OR, NOT |
| **Breakpoints** | Can be manually inserted into code by the tester in order to halt the execution of the program at specific points to inspect the values of variables | **Watch** | Usually in the format of a table and display of the values of specified fields and variables relative to the particular line the debugger is currently on. | **Steps** | Once the program is paused, the debugger allows the tester to continue the execution of the program one line at a time, effectively stepping through the program. |
| **Validation** | An automatic check performed by a computer to ensure that entered data is sensible/feasible. | **Verification** | Double checks that data has been entered correctly, data is entered twice and then compared, if they are different the user is prompted to enter again | **Character** | A single character, where a character can be any letter, digit, punctuation mark or symbol that can be typed. |
| **Syntax Error** | An error in the rules of the language. The program will not compile with syntax errors present. | **Runtime Error** | Is an error that occurs while the program is running. | **Logic Error** | The program will compile and run but the results are not what was expected e.g. the program adds 1+1 and outputs 3. |

THE Ebbsfleet ACADEMY
Aspire, Excel

## Week 1:

1. What is a variable?
2. What is a constant?
3. What is the main difference?
4. What are the arithmetic operators?

What data types would hold the following data?

5. 1. "Hello, my name is John"
6. 2. 78
7. 3. 5.423
8. 4. A
9. 5. TRUE

## Week 2 *:

Complete the trace table to determine the purpose of the algorithm. Test it with input 14 and 5.

```
1         OUTPUT ("Enter the first integer: ")
2         x ← USERINPUT
3         OUTPUT ("Enter the second integer: ")
4         y ← USERINPUT
5         z ← 0
6         WHILE x > 0
7             IF x mod 2 = 1 THEN
8                 z ← z + y
9             ENDIF
10            x ← x div 2
11            y ← y * 2
12        ENDWHILE
13    OUTPUT ("Answer =", z)
```

| x | y | x mod 2 | z | x > 0 | OUTPUT |
|---|---|---|---|---|---|
| 14 | 5 | 0 | 0 | True | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

* Algorithm pack P.66

## Week 3:

1. What is the difference between a *local* and *global* variable?
2. Define the term 'selection'
3. Define the term 'iteration'
4. Which loop uses 'count controlled' iteration?
5. Which loop uses 'condition controlled' iteration?
6. Define the term 'subroutine'
7. What is the difference between a subroutine and a function?

## Week 4:

1. Which line uses selection first?

```
Name ← USERINPUT
IF Name = "bob" THEN
    OUTPUT "Hi"
ELSE
    OUTPUT "Bye"
ENDIF
```

2. How many lines use variable assignment?

```
Name ← USERINPUT
Surname ← USERINPUT
Count ← 0
WHILE Count < 5
    OUTPUT Name, Surname
    IF Name = Surname THEN
       OUTPUT "Same names"
    ENDIF
ENDWHILE
```

```
a ← "Hello"
b ← "Hi"
```

3. What is the output of `a + b`?

4. What is the output of `len("computer")`?
5. What is the output of `char_to_code("D")`?
6. What is the output of `code_to_char(100)`?
7. What is the output of `5 MOD 3`?
8. What is the output of `5 DIV 3`?