Knowledge Organiser [redacted]

GCSE COMPUTER SCIENCE

Exam Board: OCR

Paper 1: Computer Systems

1 hour 30 minutes

Written Exam Paper

80 marks

50% of total GCSE

1.1	Computer Systems
1.2	Memory and storage
1.3	Computer Networks, Connections and Protocols
1.4	Networks Security
1.5	System Software
1.6	Ethical, Legal, Cultural & Environmental Impacts of Digital Technology

Paper 2: Computational thinking, algorithms and programming

1 hour 30 minutes

Written Exam Paper

80 marks

50% of total GCSE

2.1	Algorithms
2.2	Programming Fundamentals
2.3	Producing Robust Programs
2.4	Boolean Logic
2.5	Programming languages and IDEs

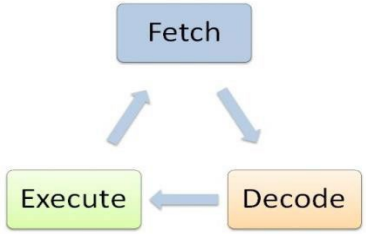
1.1 SYSTEMS ARCHITECTURE

KEY CONCEPTS

- Computer systems take data (input), process it and then output it.
- **Embedded systems** are computers built in to other devices like washing machines. They are dedicated to a single task so they are efficient.
- **Clock speed:** the number of instructions a processor can carry out per/second. Higher clock speed = faster CPU.
- **Number of Cores:** The more cores a CPU has the more instructions it can carry out at once (multitasking). More cores = faster processing.
- **Cache size:** A larger cache gives the CPU faster access to more data

FETCH – DECODE – EXECUTE CYCLE

CPU **fetches** instruction from the RAM
(Copies memory address to MAR, copies Instruction to MDR & adds 1 to PC.
CU **decodes** the instruction from the MDR Instruction is **executed** by the CU
The next instructions is fetched and
The cycle repeats.



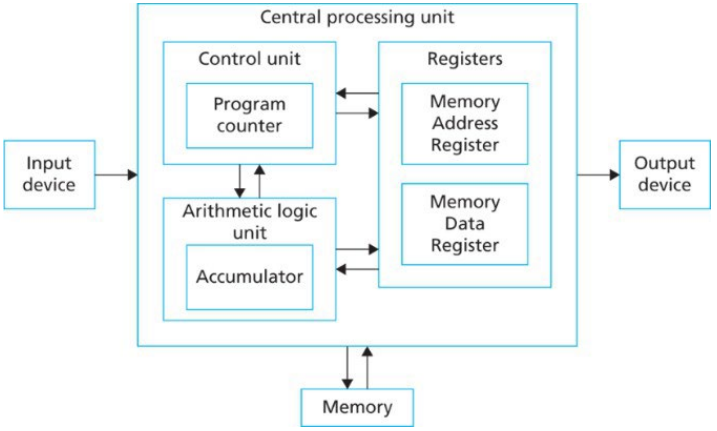
CPU COMPONENENTS AND THEIR FUNCTION

Arithmetic Logic Unit (ALU): performs the logical/arithmetic calculations.

Control Unit (CU): sends signals to coordinates the FDE cycle

Cache:very fast memory that stores regularly used data so that the CPU can access it quickly.

THE CENTRAL PROCESSING UNIT (CPU)



VON NEUMANN ARCHITECTURE (REGISTERS)

Accumulator (ACC): stores the result of calculations from the ALU.

Program counter (PC): stores the address of the next instruction to be fetched.

Memory Address Register (MAR): stores the address of the current instruction/data to be fetched.

Memory Data Register (MDR): stores the data/ instruction fetched from memory.

1.2 MEMORY and STORAGE

RANDOM ACCESS MEMORY (RAM)

- RAM is the computer's main memory that holds the data, programs and files while they are being used.
- RAM is volatile (power off = the data is lost)
- The CPU will fetch instructions from the RAM in the fetch – decode – execute cycle.
- When the RAM is full the computer uses **VIRTUAL MEMORY**. It uses part of the secondary storage as temporary RAM so that the computer can continue running (but slowly).

READ ONLY MEMORY (ROM)

- The ROM is on a chip build into the motherboard
- It contains the BIOS (boot up sequence for the computer)
- ROM is non-volatile (data still stored after power is off)

TYPES OF STORAGE

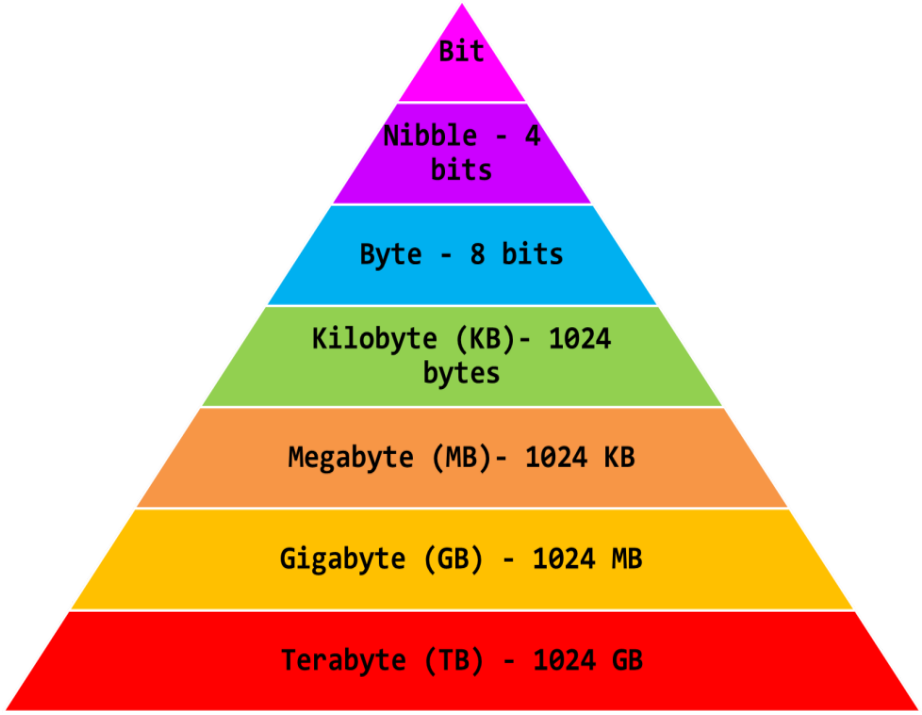
- Secondary Storage: where all data including the programs are stored when they are not being used.

Storage Type	Key Information
Magnetic	e.g. HDD, magnetic tape has moving parts, large capacity, lower cost than SSD
Solid State	e.g. SSD, Flash memory, no moving parts, more robust than HDD, faster and more expensive than HDD
Optical Storage	e.g. CDs, DVDs. Cheap, portable and fairly robust.

Storage device comparison factors: speed, cost, durability, robustness, capacity and portability.

STORAGE CAPACITY

Some storage methods such as a HDD or SSD have a large capacity (they can store lots of data. Other devices such as CDs and SD cards have smaller capacity. Measurements of capacity are shown below:



1000 instead of 1024 could be used when doing your conversion calculations, because you will not be allowed a calculator in your exam.

1.2 Data Storage

DENARY

Denary is the decimal number system that we are used to. It uses the numbers 0-9 and the column headings go up in powers of 10.

100 (Hundreds)	10 (Tens)	1 (Units)
2	3	8
2 lots of 100	3 lots of 10	8 lots of 1

BINARY

Binary uses the numbers 0 and 2. The column headings go up in power of 2:

128	64	32	16	8	4	2	1
0	1	0	0	0	1	1	1

64 + 4 + 2 + 1 = 71

HEXADECIMAL

Hexadecimal uses 0- F (A=10, B=11, C=12, D=13, E=14, F=15). The headings go up in powers of 16.

16	1
3	D
3 lots of 16	D (13) lots of 1

3* 16 = 48
D (13) * 1 = 13
48+13=61

To convert a binary number to Hexadecimal, split into 2:

8	4	2	1
0	0	1	1

= 3

8	4	2	1
1	1	0	1

= D

BINARY ADDITION

```
  1 0 0 1 0 1 0 1
+ 1 1 0 1 1 0 1 1
1 1 1 1 1 0 0 0 0
1     1 1 1 1 1
```

This binary addition gives an overflow error as the total does not fit in 8 bits (a byte).

BINARY SHIFT

A binary shift to the left multiplies the number by 2. A binary shift to the right divides it by 2. Below is an 8 bit binary number which has been shifted 2 places to the right.

Original number	1	1	0	0	1	1	0	1
Shifted number	0	0	1	1	0	0	1	1

CHARACTERS

Character sets = the characters that are recognised or represented by a computer system

ASCII = Each character is represented by an 8-bit binary number. This gives 256 different possibilities.

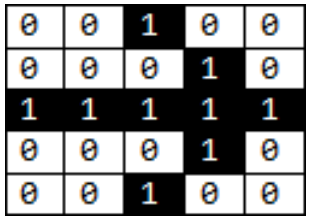
Unicode = Each letter is represented by a 16-bit binary number. This gives at least twice as many character options as ASCII and allows the character set to represent characters and symbols from all languages.

1.2 Data Storage CONTINUED

IMAGES

Images are made up of pixels
The colour of each pixel is represented by a binary number
If an image uses 1 bit to represent each colour then it will only have 2 colours:

0	0	1	0	0
0	0	0	1	0
1	1	1	1	1
0	0	0	1	0
0	0	1	0	0

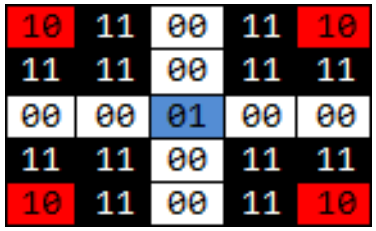


This is a 1-bit image so it uses 2 colours.

0=white and 1=black

Using more bits allows for more colour options:

10	11	00	11	10
11	11	00	11	11
00	00	01	00	00
11	11	00	11	11
10	11	00	11	10



This is a 2-bit images so it uses 4 colours.

00=white, 01=blue, 10=red, 11=black

Colour depth = the number of bits used for colour of each pixel.

Resolution = how many pixels are in a certain space – this is measured in "dots per inch". If there are more dots per inch then there are more pixels in the image so it will have a higher resolution and a better picture quality.

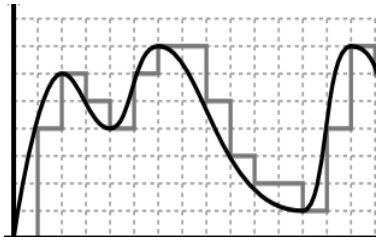
The higher the resolution or the colour depth, the more bits used, so the bigger the file size.

Metadata = the information about the image file that is stored within it. This makes sure the file is displayed correctly. It can include: the height, width, colour depth, resolution and file format as well as the time and date that the image was created.

SOUND

When sound is recorded it is an analogue signal (waves). It has to be converted to a digital signal so that it can be stored by a computer. This is done by sampling.

Sampling: The amplitude of the wave is measured at regular intervals which creates a digital representation of the wave. If samples are taken more frequently then you will end up with a more accurate sound file but it will be a larger file size.



The analogue wave is smoother and shows continuous data. The digital sampling shows the amplitude of the wave at different points.

COMPRESSION

Compression is used to make file sizes smaller. Smaller file sizes means that data will be faster to send, quicker to download (so webpages will load faster) and it will take up less storage space.

Lossy Compression: permanently removes some of the data from a file to make the file size smaller. The file – eg: an image or sound track – will be a lower quality than the original.

Lossless Compression: data is temporarily removed from the file and then put back together when it is opened. This is good for program files or documents where you do not want to lose any content but the files can only be made a little bit smaller.

1.3 COMPUTER NETWORKS

Key Terms

A network is where devices have been connected together so that they can share data and resources. Networks can be wired (Ethernet) or wireless (Wi-Fi).

Local Area Network (LAN)	Cover a small geographical area such as an office. Use their own infrastructure.
Wide Area Network (WAN)	WANs connect LANs together over a large geographical area and make use of infrastructure from telecommunications companies.
Bandwidth	The amount of data that can pass between network devices per second.
Server	A device that provides services for other devices (e.g. file server or print server).
Client	A computer or workstation that receives information from a central server.
Peer to peer Network	All of the computers in the network are equal. They connect directly to each other.
Standalone computers	A computer not connected to a network.

NETWORK HARDWARE

Network Interface Controller (NIC): built in hardware that allows a device to connect to a network.
Switches: connect devices on a LAN. Transmitting data using MAC addresses.
Router: Transmits the data (packets) between the networks, using IP addresses (e.g: the internet and your LAN).
Wireless Access Point (WAP): a switch that allows devices to connect wirelessly.
Cables: the cables in a network can be twisted pair cables, coaxial cables or fibre optic cables.

NETWORK PERFORMANCE

These factors can impact on network performance:
Bandwidth: The more bandwidth, the more data that can be transferred per second.
Number of Users: Having a lot of people using a network means lots of data is being transmitted which can slow it down because of increased traffic.
Transmission Media: Wired connections are faster than wireless. Fibre optic cables are faster than copper cables.
Wireless Factors: wireless can be affected by walls, distance, signal quality and interference from other devices.
Topology: The layout of a network can impact on its performance.

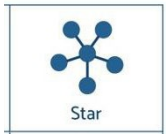
VIRTUAL NETWORKS

A virtual network is part of a LAN or WAN where only certain devices can “see” and communicate with each other usually connected remotely.

1.3 NETWORK CONNECTIONS AND PROTOCOLS

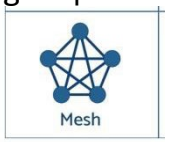
NETWORK TOPOLOGIES

A topology is the layout of a network.



Star: If the central switch fails, the whole network fails. If one device fails, the network is fine.

Mesh: Each device is connected to every other device so they can send data the fastest route. There is no single point where network can fail. Require lots of wire.



PROTOCOLS

Protocols are the rules for how devices communicate and transmit data across a network.

- **TCP/IP:** Used to send data between networks in packets.
- **Transmission Control Protocol (TCP):** Splits the data into packets and re-assembles. Checks data is sent correctly.
- **Internet Protocol (IP):** does the packet switching.
- **Hyper Text Transfer Protocol (HTTP):** for accessing websites.
- **HTTPS:** The secure version of HTTP.
- **File Transfer Protocol (FTP):** Moves files between devices.
  Post Office Protocol (POP3): Retrieves emails from server. Once you download the email the server copy is deleted.
- **Internet Message Access Protocol (IMAP):** Retrieves email from server. Email is kept on server, you see a copy.
- **Simple Mail Transfer Protocol (SMTP):** sends emails.

Standards – a set of agreed requirements for hardware and software e.g. Ethernet cable is used by computers to connect to a network.

LAYERS

Network protocols are divided into layers so that protocols with similar functions are grouped together.

Layer 4: Application	•Turn data into applications or websites •HTTP, FTP, SMTP
Layer 3: Transport	•Control the flow of data •TCP
Layer 2: Network	•Direct data packets between networks •IP
Layer 1: Data	•Sending data over a physical network •Ethernet

PACKET SWITCHING

- Data is split into packets and numbered in order.
- Each packet is send the fastest route across the internet by the routers. This means packets can take different routes and arrive out of order.
- The packet numbers are used to put them in order.
- If packets are missing a timeout message is sent
- Once all have arrived a receipt confirmation is sent to the device that sent them.

Every device has a **MAC address** so that it can be identified on a network. Eg: 98-1C-B3-09-85-15

IP addresses are used when sending data between networks. They can be static (permanent) or dynamic (different each time the device connects).
IPv4: 32 bits e.g. 37.153.62.136
IPv6: 128 bits e.g. 2001:0db8:85a3:0000:0000:8a2e:0370:7334

1.4 NETWORK SECURITY

TYPES OF ATTACK

Attack	How it works	How to prevent it
Trojan Horse	You think a legitimate piece of software is installed, this program can cause multiple problems including by creating 'back doors' for hackers to access your personal data.	Anti-malware
Virus	Could be hidden within another program, meaning the user doesn't know it exist. Can replicate itself, insert itself into other files and corrupt or delete data.	Anti-virus
Ransomware	Once on a system it encrypts all data and demands a payment for decryption	Anti-malware
Brute Force	Trial an error until a password is attacked	Strong password
Denial of Service	The network is flooded with useless data so it is too slow to use	Firewall
SQL Injection	SQL commands are typed into the input boxes on a website to access data or alter the database.	Penetration testing
Phishing	Emails with links that trick people into entering their personal information.	Looking for signs that an email is not from a real company.
Social Engineering	When a person manipulates someone else into handing over sensitive information	Policies and rules for staff about handing over data. Staff training.

NETWORK SECURITY KEY TERMS

Malware: malicious software intended to cause harm.
Penetration Testing: Organisations employ professionals to try and hack their network so that they can find areas of weakness.
User Access Levels: Different employees have different levels of access to programs, websites and data.
Encryption: data is scrambled so that it cannot be understood if intercepted. It can only be decrypted with a key.
Network Forensics: Data packets are captured as they enter the network and analysed to find out the cause of a network attack.

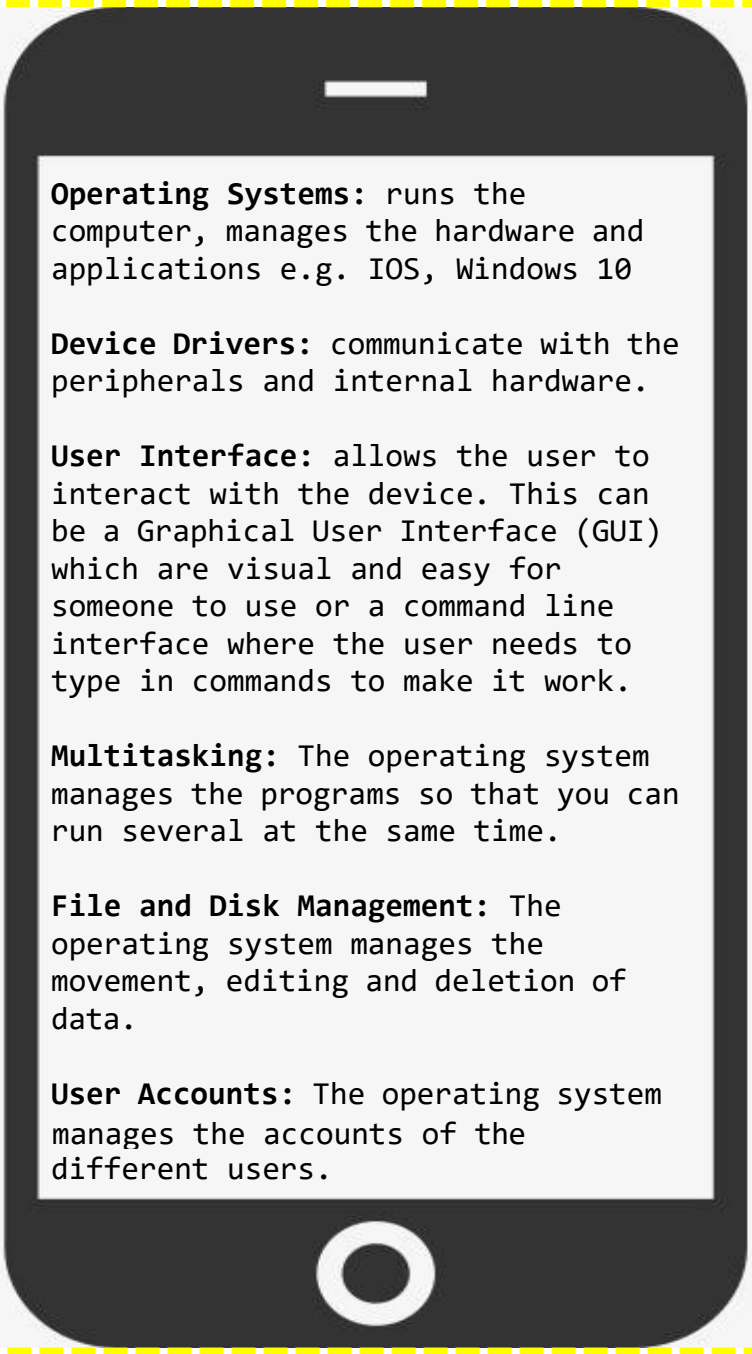
Types of Malware

Virus - attach themselves to files and copy themselves when the user copies or opens a file.

Worm - copy themselves without the user doing anything.

Trojan - malicious software pretending to be a legitimate program.

1.5 SYSTEMS SOFTWARE



Operating Systems: runs the computer, manages the hardware and applications e.g. IOS, Windows 10

Device Drivers: communicate with the peripherals and internal hardware.

User Interface: allows the user to interact with the device. This can be a Graphical User Interface (GUI) which are visual and easy for someone to use or a command line interface where the user needs to type in commands to make it work.

Multitasking: The operating system manages the programs so that you can run several at the same time.

File and Disk Management: The operating system manages the movement, editing and deletion of data.

User Accounts: The operating system manages the accounts of the different users.

Utility Software

Utilities are the programs that help maintain and configure a program. Most utility software is installed with the Operating system.

Defragmentation: Defragging a magnetic hard drive groups all of the files for each program together and all of the free space together. This makes it read and write quicker.

Compression: reduces the size of large files so that they take up less space. Files then need to be extracted before they are used.

Encryption: scrambles the data to protect it so that if someone else gets hold of it they cannot access it.

Open Source and Proprietary Software

Open Source	Proprietary
It's usually free and the source code is available so it can be adapted for individual needs	Usually has to be paid for Only the compiled code is released so it cannot be edited

1.5 ETHICAL, LEGAL, CULTURAL & ENVIRONMENTAL IMPACTS

Ethical

- Ethics is about what is considered right and wrong by society.
- If a company does not behave in an ethical way it might make their customers lose trust in them.
- Issues such as cyberbullying, trolling and the use of social media can raise ethical issues.
- **Privacy:** Users trust companies to keep their data private so companies need to take care of it
- **Censorship:** is when a country or organisation controls what people can access on the internet.
- **Surveillance:** surveillance is when someone is monitored using technology.

Legal

- **Data Protection Act:** controls how personal data is used. Eg: it has to be accurate and up to date, kept secure, should not be kept longer than needed
- **Freedom of information Act:** gives the public the right to see information about public organisations
- **Computer Misuse Act**: makes it illegal to hack a network or create a virus.
- **Copyright, Designs & Patents Act:** protects things you have created from being used without permission
- **Creative Commons:** lets people release their work to be used and shared legally and sometimes modified.

Stakeholders:

The people or groups affected by a particular situation

Environmental

- Computing devices contain raw materials
- Devices use lots of energy when turned on
- **Ewaste** is when we throw away devices because they are broken or because we want to upgrade
- Ewaste can lead to pollution
- The **Waste Electric and Electronic Equipment (WEEE)** directive has rules for how devices should be disposed so that they're recycled/disposed of safely
- Devices can also have a positive impact on the environment – eg video calls rather than travelling a long distance causing pollution.

Cultural

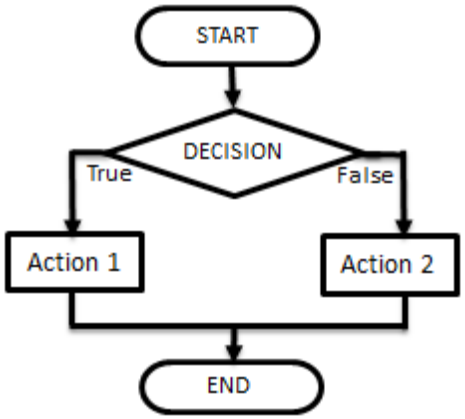
- One cultural issue in computing is the **Digital Divide.** Some people do have access to technology, others don't
- Not having access to technology can be a disadvantage as it limits access to information, online learning, online banking, communication etc.
- The digital divide can be due to people not having enough money to buy devices or due to living in places without internet access, or not having the skills to use the technologies available.
- Technology has also impacted how businesses run as many now use online shops and services

2.1 ALGORITHMS

COMPUTATIONAL THINKING

Abstraction	•Focussing on just the important details of a problem
Decomposition	•Breaking a problem down into smaller parts so that it is easier to solve
Algorithmic thinking	•creating a step by step solution to a problem

FLOWCHART



PSEUDOCODE

```
START
IF the Decision = TRUE THEN:
    Perform Action 1
ELSE
    Perform Action 2
ENDIF
END
```

- Universal language, planning phase before actual coding in for e.g. python
- Work out where you need inputs, outputs, decisions, loops and variables.

SEARCHING ALGORITHMS

To find an item in a list, computers need to use a searching algorithm. A linear search and binary search are both examples of sorting algorithms.

Linear Search: Checks each item in the list one by one until it finds what it is looking for
+ Simple, list doesn't need to be ordered
- Not efficient, takes time with lots of data

Binary Search: Finds the middle item in an ordered list by doing (n+1)/2. IF the middle item is what it is searching for it stops. If not, it compares the item you are searching for to the middle item so that it knows whether to look in the first half or second half of the list. Then it repeats these steps until the item is found
+ More efficient than a linear search
- Only works on an ordered list, complex to program

SORTING ALGORITHMS

Sorting algorithms sort items into an ordered list.

Bubble Sort: Checks the first two items in a list, swaps them if they are in the wrong order and then moves onto the next two items and repeats the process. Once it has passed through the list once it goes through again until none of the items need swapping. + Simple. – Takes a long time

Merge Sort: Finds the middle item (n+1)/2 and splits the list in half. Repeats this step until the list is split into individual items (sub-lists). It them merges (joins) the sublists in pairs. Each time the sublists are paired they are sorted into the correct order. + Efficient – Slow

Insertion Sort: Looks at the second item in a list and compares it to the items that are in front of it, then inserts it into the right place. It then moves to the next item in the list and repeats these steps. + Quick for sorting small lists – slow with long lists

2.2 PROGRAMMING FUNDAMENTALS

DATA TYPES

Data Type	Definition
String	Text eg: "Hello"
Integer	Whole number eg: 32
Float/Real	Decimal number eg: 1.2
Boolean	Two values eg: true or false
Character	A single character eg: b

VARIABLES AND CONSTANTS

Variable – A value which may change while the program is running. Variables can be local or global.

Local Variable – a variable which can only be used within the structure they are declared in.

Global Variable – a variable which can be used in any part of the code after they are declared

Constant – A value which cannot be altered as the program is running.

OPERATORS

Operator/Function	Definition
Exponentiation	Raises a number to a power eg: 2**3 OR 2 ^3 (=2^3)
Quotient/DIV	Gives the whole number after a division e.g. 5 DIV 2 = 2
Remainder/MOD	Gives the remainder part of a division e.g. 5 MOD 2 = 1
==	Is equal to
!= or <>	Is not equal to
<	Is less than
>	Is more than

ARRAYS

One-Dimensional Arrays- this is like a list.
In this example an array has been created called students. The list can hold 3 items (as shown).

```
array students [3]
students [0] = "Bob"
students [1] = "Dave"
students [2] = "Bob"
```

This command would print the second item (1) From the array. It would print "Dave".

```
print(students[1])
```

Two-Dimensional Arrays – these are lists within lists (like a table)

```
Grades=[["Bob", "22%", "44%"], ["Dave", "85%", "100%"]]
```

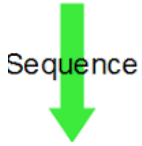
	0	1	2
0	Bob	22%	44%
1	Dave	85%	100%

The code above creates the 2D array. The code Below would output:
"Bob's first test score was 22%"

```
print("Bob's first test score was " + Grades [0, 1]
```

2.2 PROGRAMMING FUNDAMENTALS CONTINUED

PROGRAMMING CONSTRUCTS



A Sequence is when there are programming steps that are carried out one after another.

Selection is where there are different paths in your code eg: IF, ELIF, ELSE

Iteration is when there is repetition (loops) in code. This could be a WHILE loop (do something WHILE a condition is met) or a FOR loop (do something for a set number of times)

This count-controlled loop would print "Hello World" 8 times.:

```
for i=0 to 7
      print ("Hello")
next i
```

These condition controlled loops would check if a password's correct:

```
while answer != "letmein123"
      answer=input("Enter password")
endwhile
```

```
do
      answer=input("Enter password")
until answer=="letmein123"
```

STRING MANIPULATION

```
0   1   2   3
W   o   r   d
```

The characters in a string are numbered starting with position 0.

Function	Purpose
x.length	Gives the length of the string
x.upper	Changes the characters in the string to upper case
x.lower	Changes the characters in the string to lower case
x[i]	Gives the character in position i. Eg: x[2] = "r"
x.substring(a,b)	Gives the characters from position a with length b. Eg: x.subString(1,2) = or
+	Joins (concatenates) two strings together

FILE HANDLING

Myfile=openRead("myfile.text")	Opens the file in read mode
Myfile=openWrite("myfile.text")	Opens the file in write mode
Myfile.writeLine ("Hello")	Writes a line to the file
Line1=myfile.readLine()	Reads one line of the file
Myfile.close()	Closes the file
endOfFile()	Used to determine the end of a file

IF/ELSE AND SWITCH/CASE FOR SELECTION

IF ELSE

```
If choice == "a" then
   print("You chose A")
elseif choice=="b" then
   print("You chose B")
else
   print("Unrecognised choice")
```

SWITCH/CASE

```
Switch entry:
   case "A":
      print("You chose A")
   case "B":
      print("You chose B")
   default:
      print("Unrecognised choice")
```

2.2 PROGRAMMING FUNDAMENTALS CONTINUED

SUB PROGRAMS

Procedures are a set of instructions stored under a name so that you can call the procedure to run the whole set of instructions.
A **function** is like a procedure but always returns a value.
Parameters are variables used to pass values into a function or procedure.

A procedure **with** parameters	A procedure **without** parameters
procedure intro (name) print("Hello " +name) print("Welcome to the game") endprocedure	procedure intro () print("Hello") print("Welcome to the game") endprocedure

SQL (Structured Query Language)

SQL is the language used to manage and search databases.

Commands	Example	What it does
SELECT FROM	SELECT name, age FROM students	Displays the name and age of everyone in the students table
WHERE	SELECT name FROM students WHERE gender=male	Displays the name of everyone in the students table who's gender is male
LIKE	SELECT name FROM students WHERE name LIKE "% Smith"	Displays the students' names that end with Smith.
AND	SELECT name FROM students WHERE gender=male AND attendance > 90	Displays the students who are male and have an attendance of more than 90.
*	SELECT * from students	Selects all of the fields from the students table

RECORDS

Records are a data structure used to store a collection of data. They can store information of different data types.
Field = each item in a record is a field.
Each field has a name and data type.

A record can be created like this:

```
record students
    int student_number
    string student_name
    bool passed_test
endrecord
```

Data can be assigned using variables:

```
Student1=students(1,"Bob Jones", True)
Student2=students(2,"Steve Smith", False)
Student3=students(3,"Sally Roberts", True)
```

The whole record can be accessed using the variable name:

```
print(Student1)
```

(1, "Bob Jones", True)

or part of a record can be accessed:

```
print(Student3.student_name)
```

Sally Roberts

2.3 PRODUCING ROBUST PROGRAMS

DEFENSIVE DESIGN

Programmers try to protect their programs by testing them to reduce the number of errors, predicting how users might misuse their program and trying to prevent it and making sure their code is well maintained.

Input Sanitisation – removes any unwanted characters that have been entered into a program

Input Validation – Checks if the data meets certain criteria before passing it through the program. The following validation checks can be used:

Presence check	Checks that data has been entered
Length check	Checks the data is the correct length
Range check	Checks the data is within a set range
Format check	Checks it's in the correct format (Eg:dd/mm/yy)
Check digit	Checks numerical data is entered correctly
Look-up table	Checks against a table of accepted values

Authentication – Where a program confirms the identity of a user before giving them access to the full program. This could be done through usernames and passwords.

Maintainability – Code that has been well maintained is easy to edit without causing errors. A well maintained code will have **comments** to help other programmers understand the code, as well as **appropriate names for variables and sub programs**, and **indentation** so that it is easy for programmers to see the flow of the program. Global variables should only be used where necessary so that they don't impact on the rest of your code.

TESTING

A program should be tested to check for any errors.

Final Testing – The program goes is tested once at the end of development. Everything is tested in one go.

Iterative testing - a program is tested and then changes are made as it goes through the development cycle again. It may go through this process a few times to make sure it is exactly what the customer wants.

Test data can fit into 3 different categories:

Normal	Data which is likely to be entered into the program and should be accepted
Extreme/ boundary	Data on the limit of what should be accepted
Erroneous	Data that should not be accepted

TYPES OF ERROR

A program should be tested to check for any errors.

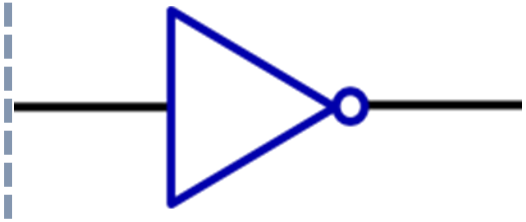
Syntax Error – something which doesn't fit the rules or grammar of the programming language.

Logic Error – the program runs but not as expected. Eg: < user instead of >.

2.4 COMPUTATIONAL LOGIC

NOT GATE

A NOT gate takes an input and outputs the opposite.



Input	Output
0	1
1	0

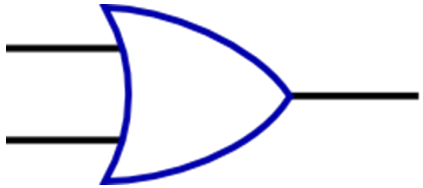
AND GATE

For an AND gate to give an output of 1, both inputs must be 1.

Input A	Input B	Output
0	0	0
1	0	0
0	1	0
1	1	1

OR GATE

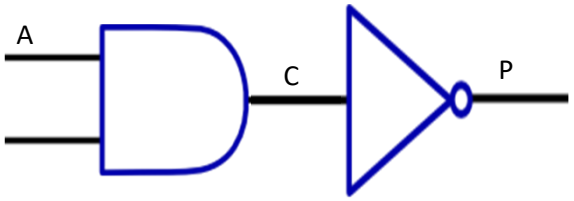
For an OR gate to give an output of 1, either inputs must be 1.



Input A	Input B	Output
0	0	0
1	0	1
0	1	1
1	1	1

COMBINED GATES

Logic gates can be combined:



A	B	C	P
0	1	0	1
1	0	0	1
1	1	1	0
0	0	0	1

LOGIC EXPRESSIONS

The table below shows the logic gate expressions and notations that you need to know:

Gate	Expression	Notation
NOT	NOT A	¬A
AND	A AND B	A ∧ B
OR	A OR B	A ∨ B

WHY COMPUTERS USE BINARY

Computers use 1s and 0s to represent the flow of electricity in their circuits.

0 = off
1 = on

Bit = a single bit (0 or 1)
Nibble = 4 bits
Byte = 8 bits
Kilobyte = 1000 bytes
Megabyte = 1000 kilobytes
Gigabyte = 1000 megabytes
Terabyte = 1000 gigabyte
Petabyte = 1000 terabytes

2.5 Programming Languages and IDEs

HIGH LEVEL LANGUAGES

- Eg: Python, Java etc
- Each instruction in a high-level code represents many machine code instructions.
- The code will work on many different computers and processors
- Data can be stored in different structures like lists and arrays
- The code is easy to read and understand
- The code has to be converted into machine code for the computer to understand it
- Programs will be less memory efficient as there is no control over what the CPU does

LOW LEVEL LANGUAGES

- Eg: Machine code (binary) and assembly language
- Each instruction only represents one instruction of machine code
- Low level languages are written for one particular machine or processor
- To store data the programmer needs to understand how the CPU manages memory
- Low level code is difficult to read and understand
- Machine code can be executed without translators
- Programs are more memory efficient as you control what the CPU does

TRANSLATORS

High level languages have to be translated to machine code for the computer to understand them.

Assemblers – turn assembly language into machine code

Compilers – Translate all of the code in on go to create an executable file. A compiler can take a long time but the final code runs quickly and gives a list of errors for the entire program.

Interpreters – Translates the code one instructions at a time. This means the program will run more slowly. No executable file is created so the code will need to be translated every time it runs. The interpreter will stop after each error which is helpful when debugging

IDE'S (INTEGRATED DESIGN ENVIRONMENTS)

IDE's help programmers develop their code. They have a range of features to do this:

Editors – the area which the code is written in. Includes line numbers and colour coding for different features of the code (variables, comments etc)

Run Time Environment – Lets the programmer run the code quickly to test it for errors

Error Diagnostics – includes diagnostic tools to help find and solve errors

A Translator – to translate the code into machine code

Breakpoints – Stop the program on certain lines so that information up to that point can be gathered.