

Cambridge AS & A Level · Computer Science 9618

Computer Science 9618 formulas and definitions

All 207 formulas, facts and definitions to learn, by syllabus topic. Formulas the exam gives you are left out.

Read online at pastopic.com/cambridge/computer-science-9618/formulas; each topic heading links to its full note.

© 2026 Pastopic. Free for personal study. Please share the link, not the file.

Information representation

n bits give 2^n different values; the least bits for N values is the smallest n with $2^n \geq N$

1 KiB = 2^{10} = 1024 bytes, MiB = 2^{20} , GiB = 2^{30} , TiB = 2^{40} bytes

1 kB = 10^3 bytes, MB = 10^6 , GB = 10^9 , TB = 10^{12} bytes

1 byte = 8 bits; 1 nibble = 4 bits = 1 hex digit

Hex place values: 4096, 256, 16, 1; A–F = 10–15

Two's complement: MSB worth -2^{n-1} ; negate by flipping all bits and adding 1

n -bit ranges: unsigned 0 to $2^n - 1$; two's complement -2^{n-1} to $2^{n-1} - 1$

BCD: each denary digit as 4 bits; valid nibbles 0000–1001

Bitmap size (bits) = width $\times$ height $\times$ colour depth

Sound size (bits) = sampling rate $\times$ sampling resolution $\times$ time in seconds

Bits $\div 8$ = bytes; then $\div 1000$ per step for kB, MB, GB or $\div 1024$ per step for KiB, MiB, GiB

ASCII 7 bits (128 characters); extended ASCII 8 bits (256); Unicode up to 32 bits (16 in most answers)

Communication

IPv4: 4 groups $\times$ 8 bits = 32 bits, denary, each group 0–255, full stops

IPv6: 8 groups $\times$ 16 bits = 128 bits, each group 4 hex digits (0000–FFFF), colons; one run of zero groups can become ::

Number of addresses: IPv4 $2^{32} \approx 4.3 \times 10^9$; IPv6 $2^{128} \approx 3.4 \times 10^{38}$

IP address in a subnet = network ID + host ID; mask bits 1 = network ID, 0 = host ID

1 Mbps = 10^6 bits per second; 1 byte = 8 bits

Data (bits) = bit rate (bits per second) $\times$ time (seconds)

$$\text{Time (seconds)} = \frac{\text{data in bits}}{\text{bit rate}}$$

Smooth streaming needs broadband speed $\geq$ the stream's bit rate

Hardware

NOT: output = 1 only when the input = 0

AND: output = 1 only when both inputs = 1; NAND is the reverse (output = 0 only for 1, 1)

OR: output = 1 when at least one input = 1; NOR is the reverse (output = 1 only for 0, 0)

XOR: output = 1 only when the inputs are different

Truth table rows = 2^n for n inputs: 3 inputs $\rightarrow$ 8 rows, in binary order 000 to 111

Sum of products: one AND term per output-1 row (NOT on each 0 input), terms joined by OR

RAM: volatile, read/write, current programs and data. ROM: non-volatile, start-up instructions and firmware

SRAM: flip-flops, no refresh, faster, dearer, cache. DRAM: capacitors, refreshed, cheaper, denser, main memory

PROM: write once. EPROM: UV, removed, whole chip. EEPROM: voltage, in place, byte by byte

Control system: sensor $\rightarrow$ processor compares with set value $\rightarrow$ actuator $\rightarrow$ feedback (output changes next input)

Processor fundamentals

Fetch: $\text{MAR} \leftarrow [\text{PC}]; \text{PC} \leftarrow [\text{PC}] + 1; \text{MDR} \leftarrow [[\text{MAR}]]; \text{CIR} \leftarrow [\text{MDR}];$ then decode and execute

$[X]$ = contents of X; $[[\text{MAR}]]$ = contents of the memory location whose address is in the MAR

Address bus one way (from CPU); data and control buses both ways

n address lines address 2^n memory locations

Immediate: operand is the value. Direct: operand is the address. Indirect: operand holds the address.

Indexed: operand + IX. Relative: offset from a base address

Logical shift fills with 0; arithmetic right shift copies the sign bit; cyclic shift wraps round

Left shift by n : $\times 2^n$; right shift by n : $\div 2^n$, remainder lost (unsigned)

AND with 0 clears, AND with 1 tests; OR with 1 sets; XOR with 1 flips

Two-pass assembler: pass 1 builds the symbol table; pass 2 replaces labels and generates object code

System software

Five key management tasks: memory, file, security, hardware (input/output/peripherals), process

Six utilities: disk formatter, virus checker, defragmentation, disk contents analysis / disk repair, file compression, back-up

Three translators: assembler, compiler, interpreter

IDE features: coding (context-sensitive prompts), error detection (dynamic syntax checks), presentation (prettyprint, expand/collapse), debugging (single stepping, breakpoints, watch and report windows)

Security, privacy and data integrity

Security: protect from loss, corruption, damage · Privacy: only authorised people can access · Integrity: accurate, complete, consistent, up to date

System measures: user accounts, passwords, biometrics and other authentication, digital signatures, firewall, anti-virus, anti-spyware, encryption

Data measures: encryption, access rights (none, read-only, read and write)

Threats: virus, spyware, hacker, phishing, pharming

Digital signature: hash → digest → encrypt with sender's private key → send both → decrypt with sender's public key → hash again → compare

Validation (reasonable, by the computer) vs verification (matches the original)

Validation checks: range, limit, presence, length, format, existence, check digit (type also accepted)

Verification on entry: visual check, double entry · on transfer: parity byte, parity block, checksum

Parity: even parity makes the total number of 1s even; odd parity makes it odd

Ethics and ownership

Professional ethical bodies: BCS and IEEE; you join as an individual, not as an employee

Benefits of joining: written guidelines, recognition and reputation, legal advice and support, training, networking

Code of conduct: clear expectations and consequences, good practice, professional workplace, reputation and public trust

Ethical duties: public, client and employer, product, judgement, management, profession, colleagues, self

Copyright: legal right of ownership of intellectual property; makes copying illegal without permission; right of legal redress

Software licence: legal agreement setting out the rights and terms of use

Free software (FSF): free of restrictions, not necessarily free of charge; source code included; may run, change and share

Open source (OSI): source available; may be freely copied, distributed and adapted

Shareware: free trial (limited time or features), then pay

Commercial: paid, no source code, can't legally change or share, install limits, support and updates

AI chain: capture → analyse patterns → compare with learned or stored patterns → most probable match → act → learn

AI impacts: social, economic, environmental (and ethical)

Databases

CREATE DATABASE Name;

CREATE TABLE T(Field TYPE, ..., PRIMARY KEY (Field), FOREIGN KEY (Field) REFERENCES Other(Field));

ALTER TABLE T ADD Field TYPE; and ALTER TABLE T ADD FOREIGN KEY (Field) REFERENCES Other(Field);

SELECT fields FROM T WHERE condition ORDER BY field [ASC or DESC];

Join: FROM A INNER JOIN B ON A.Key = B.Key or FROM A, B WHERE A.Key = B.Key

COUNT(field), SUM(field), AVG(field), named with AS Title; GROUP BY field for "each"

INSERT INTO T (fields) VALUES (values); UPDATE T SET F = v, G = w WHERE ...; DELETE FROM T WHERE ...;

Text in quotes "C101"; numbers and TRUE/FALSE without quotes; dates as #31/12/2025#

1NF: no repeating groups, atomic fields, primary key

2NF: 1NF and no partial dependencies

3NF: 2NF and no non-key (transitive) dependencies; no many-to-many relationships left

1:M is built with the "one" side's primary key as a foreign key in the "many" side; M:M needs a linking table

Algorithm design and problem-solving

Random whole number from Low to High inclusive: $\text{INT}(\text{RAND}(\text{High} - \text{Low} + 1)) + \text{Low}$

Last digit of a whole number: $N \bmod 10$; remove the last digit: $N \text{ DIV } 10$

Add a digit D to the end of a whole number: $N * 10 + D$

One character of a string: $\text{MID}(S, \text{Index}, 1)$, for Index from 1 to $\text{LENGTH}(S)$

X is even: $X \bmod 2 = 0$

Inclusive range: $X \geq \text{Low} \text{ AND } X \leq \text{High}$; outside it: $X < \text{Low} \text{ OR } X > \text{High}$

$\text{NOT } (A \text{ OR } B) = \text{NOT } A \text{ AND } \text{NOT } B$; $\text{NOT } (A \text{ AND } B) = \text{NOT } A \text{ OR } \text{NOT } B$

FOR: number of repeats known; WHILE: tested first, may run 0 times; REPEAT ... UNTIL: tested last, runs at least once

Function header: `FUNCTION Name (Param : TYPE) RETURNS TYPE ... ENDFUNCTION`; a procedure has no RETURNS

Data types: INTEGER, REAL, CHAR, STRING, BOOLEAN, DATE

Flowchart symbols: terminator, input/output (parallelogram), process (rectangle), decision (diamond), subroutine, flow line

Data types and structures

Data types: INTEGER (whole number), REAL (can have a fraction), CHAR (one character, 'A'), STRING ("text"), BOOLEAN (TRUE/FALSE), DATE (21/10/2026)

Record type: `TYPE Name ... DECLARE Field : Type ... ENDTYPE`; a field is reached with dot notation, `Var.Field` or `Array[Index].Field`

1D array: `DECLARE Name : ARRAY[Lower:Upper] OF Type`

2D array: `DECLARE Name : ARRAY[L1:U1, L2:U2] OF Type`; an element is `Name[Row, Column]`

Elements in one dimension of an array = upper bound – lower bound + 1; a 2D array has rows × columns elements

Text files: `OPENFILE File FOR READ, WRITE or APPEND; READFILE File, Var; WRITEFILE File, Data; EOF(File); CLOSEFILE File`

WRITE mode creates the file and erases what was there; APPEND adds to the end; READ only reads

Bubble sort on n items: $n - 1$ passes always sort it; a sort that stops only after a pass with no swaps may need one more pass to confirm the order

ADT: a collection of data and a set of operations on that data

Stack: last in, first out (LIFO), one pointer to the top; queue: first in, first out (FIFO), front and end pointers

Circular queue: when a pointer passes the upper bound it wraps around to the lower bound

Linked list: nodes of data + pointer, a start pointer, a free list pointer, and a null pointer that can never be an index

Programming

Constant and variable: `CONSTANT Name = Value; DECLARE Name : Type; assignment Name ← Expression`

Selection: `IF ... THEN ... ELSE ... ENDIF; CASE OF Var ... Value : ..... OTHERWISE : ... ENDCASE` (first matching clause runs)

Loops: `FOR Var ← A TO B [STEP S] ... NEXT Var; WHILE Cond ... ENDWHILE; REPEAT ... UNTIL Cond`

Procedure: `PROCEDURE Name (P : Type, ...) ... ENDPROCEDURE`, used with `CALL Name (...)`

Function: `FUNCTION Name (P : Type, ...) RETURNS Type ... RETURN Value ... ENDFUNCTION`, used inside an expression

`BYVAL` (default) passes a copy; `BYREF` passes the caller's variable, so changes are kept after the call

Random whole number from 1 to N: `INT (RAND (N) ) + 1`

Last digit of a whole number: `Num MOD 10`; the number without its last digit: `Num DIV 10`

Any substring from LEFT and RIGHT: `MID (S, x, n) = RIGHT (LEFT (S, x + n - 1), n)`

Software development

Life-cycle stages in order: analysis, design, coding, testing, maintenance

Analysis produces the requirements specification; design produces structure charts, algorithms, data structures, interface layouts and the test plan

Waterfall: stages done once each, in order, each finished before the next; down arrows pass each stage's output on, up arrows return to the previous stage

Iterative: the program is built in cycles, each adding features to a working version

RAD: quick prototypes, repeated client feedback, modules built in parallel to time limits, reuse of code

Structure chart: curved arrow means repeated calls; diamond means one call chosen at run time; two-headed arrow means a `BYREF` parameter

Structure chart arrows: down is a parameter, up is a returned value (`FUNCTION`); filled circle is a `BOOLEAN` flag

State-transition arrow label: the input, then a bar, then the output (input alone if there is no output)

Syntax error: grammar rules broken, found by the compiler; logic error: runs but wrong result; run-time error: illegal operation while running (crash or freeze)

Test data: normal (well inside), extreme (largest and smallest valid), boundary (each edge from both sides), abnormal (should be rejected)

White-box: every path through the code; black-box: inputs and outputs only, code not seen

Stub: dummy module that returns a known value or outputs a message in place of an unfinished module

Integration: tested modules combined and tested together; alpha: by the developers; beta: by real users before release; acceptance: by the client against the requirements

Maintenance: corrective fixes errors, adaptive meets new requirements or environments, perfective improves performance or usability

IDE debugging order: breakpoint, then single-stepping, while a watch window shows the variables

Data representation

Enumerated type: `TYPE <name> = (<value1>, <value2>, ...)`

Pointer type: `TYPE <name> = ^<data type>; address of x: ^x; value pointed to: P^`

Record type: `TYPE <name> ... DECLARE <field> : <type> ... ENDTYPE`

Record variable: `DECLARE <var> : <record type>; field: <var>.<field> ← <value>`

Set: `TYPE <name> = SET OF <type> then DEFINE <set> (<values>) : <name>`

Floating-point value = $M \times 2^E$ (mantissa a two's complement fraction, point after the first bit; exponent a two's complement integer)

Mantissa place values: $-1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots$; exponent with e bits: $-2^{e-1}, \dots, 2, 1$

Normalised mantissa: positive starts 01, negative starts 10

Moving the mantissa one place left: exponent -1

m -bit mantissa, e -bit exponent: largest = $(1 - 2^{-(m-1)}) \times 2^{2^{e-1}-1}$

Smallest positive normalised = $\frac{1}{2} \times 2^{-2^{e-1}}$

More mantissa bits → more precision; more exponent bits → more range

Hashing: address = key MOD n (for n addresses 0 to $n - 1$)

Communication and internet technologies

Protocol = an agreed set of rules for communication, so different hardware and software can exchange data

TCP/IP layers, top to bottom: Application, Transport, Internet, Link

Each layer only passes data to the layer directly above or below it; sending goes down the stack, receiving goes up

Transport: segments, sequence numbers, ports, end-to-end delivery, flow control, resending

Internet: IP addresses in the header (datagrams), routing

Link: frames, MAC addresses, the physical link to the next device

HTTP(S): web pages · FTP: files · SMTP: send email (push) · POP3 and IMAP: receive email (pull) ·

BitTorrent: peer-to-peer file sharing

POP3 downloads to one device; IMAP keeps mail on the server and synchronises every device

BitTorrent: torrent file (metadata) → tracker → swarm of peers → pieces → seeds

Circuit switching: dedicated path set up first, whole bandwidth, same route, in order

Packet switching: packets with headers, independent routes, reordered by sequence number, lost packets resent

Hop count: reduced by 1 at each router; the packet is discarded at 0

Hardware and virtual machines

RISC: few, simple, fixed-length instructions; one clock cycle each; few addressing modes; many general-purpose registers; hard-wired control unit; easy to pipeline; emphasis on software

CISC: many complex, variable-length instructions; several clock cycles each; many addressing modes and formats; microprogrammed control unit; uses cache; emphasis on hardware

Pipeline stages: IF, ID, OF, IE, WB; n instructions in a 5-stage pipeline take $n + 4$ cycles (not $5n$)

SISD, SIMD, MISD, MIMD: single or multiple **instruction** streams, then single or multiple **data** streams

Half adder: $S = A \oplus B, C = A.B$

Full adder: $S = A \oplus B \oplus C_{in}$ (1 for an odd number of 1s); $C_{out} = 1$ when two or more inputs are 1

SR: 10 sets, 01 resets, 00 holds, 11 invalid. JK: the same, but 11 toggles, and it is clocked

De Morgan: $\overline{A.B} = \overline{A} + \overline{B}$ and $\overline{A + B} = \overline{A}.\overline{B}$

$A + \overline{A} = 1, A.\overline{A} = 0, A + A.B = A, A + \overline{A}.B = A + B$

K-map order 00, 01, 11, 10; loops of 1, 2, 4, 8; wrap around the edges

System software

Process states: running, ready, blocked. Ready $\rightarrow$ running (given the processor); running $\rightarrow$ ready (time slice ends or pre-empted); running $\rightarrow$ blocked (waits for I/O or event); blocked $\rightarrow$ ready (event complete)

FCFS: arrival order, to the end. SJF: shortest burst first, to the end (non-pre-emptive). SRT: shortest time left, pre-emptive. Round robin: equal time slices in turn, pre-emptive

Kernel as interrupt handler: receive, check priority, save state, find ISR in the interrupt dispatch table, run it, restore state

Paging: fixed-size pages and frames, page table, page number + offset. Segmentation: variable-size logical segments, segment map table, segment number + offset

Page replacement: FIFO (oldest page) or LRU (least recently used page)

Disk thrashing: pages swapped in and out so often that more time is spent swapping than processing

Compilation: lexical analysis $\rightarrow$ syntax analysis $\rightarrow$ code generation $\rightarrow$ optimisation

BNF: $::=$ is defined as, | or, <name> non-terminal; repetition by recursion, such as <digits> $::=$ <digit> | <digit><digits>

RPN with a stack: push operands; for an operator pop two, apply (second popped on the left), push the result

Security

Plaintext $\rightarrow$ (encrypt with a key) $\rightarrow$ ciphertext $\rightarrow$ (decrypt with a key) $\rightarrow$ plaintext

Symmetric: one shared secret key encrypts and decrypts; fast, shorter keys (128/256 bits), but the key must be shared

Asymmetric: a matching public/private key pair; one encrypts, only the other decrypts; slower, longer keys (2048 bits), only the public key is shared

Private message: encrypt with the **receiver's public key**, decrypt with the **receiver's private key**

Verified message: encrypt with the **sender's private key**, decrypt with the **sender's public key**

Digital signature = message digest (hash of the message) encrypted with the sender's private key

Check: decrypt signature with sender's public key → digest 1; hash received message → digest 2; equal means unaltered

Digital certificate: issued by a CA; holder's name and public key, serial and version numbers, validity dates, issuer, CA's signature

Quantum cryptography: photons over fibre; eavesdropping detected because measuring changes the state

SSL/TLS: encryption, authentication, integrity; handshake protocol + record protocol

TLS handshake: certificate sent → verified → server's public key → algorithms agreed → symmetric session keys → secure session

Artificial intelligence (AI)

Graph: **nodes** (vertices) joined by **edges**; edges can be **weighted** (distance, time, cost)

Dijkstra's: shortest distance from the start to **every** node; set start = 0, others = ∞ ; visit the smallest unvisited; update a neighbour only if the new total is smaller

A*: $f = g + h$, where g = actual cost from the start, h = heuristic (estimated cost to the goal); expand the lowest f

A* uses a heuristic to find the shortest path between **two** nodes, exploring fewer nodes; Dijkstra's explores all routes

ANN: input layer → hidden layers → output layer; weighted connections; weights adjusted by training

Back propagation: compare actual with expected output → error gradient (calculus) → feed error back → adjust weights → repeat

Supervised = **labelled** data, predicts outputs; unsupervised = **unlabelled** data, finds hidden patterns/clusters; reinforcement = **rewards and penalties**, trial and error

Deep learning = machine learning with ANNs that have **many hidden layers**; best with large data sets

Regression: predicts a **numerical value** by fitting a line or curve that minimises the errors

Computational thinking and problem-solving

Linear search: worst case n comparisons, $O(n)$

Binary search: worst case $\lfloor \log_2 n \rfloor + 1$ comparisons, $O(\log_2 n)$; data must be sorted

Bubble sort and insertion sort: $O(n^2)$ worst case, $O(n)$ if already sorted, space $O(1)$

Stack push and pop, queue enqueue and dequeue: $O(1)$

Middle index: $\text{Mid} \leftarrow (\text{Lower} + \text{Upper}) \text{ DIV } 2$

Circular wrap: IF $\text{Pointer} > \text{MaxSize}$ THEN $\text{Pointer} \leftarrow 1$ (or 0 for a 0-based array)

Stack: LIFO, one end, one moving pointer (Top)

Queue: FIFO, two ends, two moving pointers (Front, Rear)

Recursion: general case, base case, moves towards the base case

Further programming

Immediate: operand is the value. Direct: operand is the address. Indirect: operand is the address of the address. Indexed: operand + IX. Relative: offset from a base address

Class: template of attributes and methods. Object: an instance of a class

Encapsulation: data and methods in one class, attributes `PRIVATE`, reached through methods

Getter returns an attribute's value; setter changes it

Inheritance: subclass takes the superclass's attributes and methods and can add or override them

Polymorphism: same method name, redefined to behave differently in subclasses

Containment (aggregation): a class holds objects of another class ("has a")

Fact $f(a, b)$., rule $\text{head}(X) \text{ IF } p(X) \text{ AND NOT } q(X)$., goal $f(X, b)$ returns $X = \dots$

`OPENFILE ... FOR READ / WRITE / APPEND / RANDOM, READFILE, WRITEFILE, EOF(),`
`CLOSEFILE`

`SEEK file, position, GETRECORD file, variable, PUTRECORD file, variable`

Exception: unexpected run-time event; causes: programming, user, hardware errors
