THE GRIPHAN ONLINE SCHOOL

UNIVERSITY LIBRARY

Computer Science Foundations

How computers work, and how to think like a computer scientist

Nji Griphan

griphan.48hourrfprescue.com

Computer Science Foundations

How computers work, and how to think like a computer scientist

Nji Griphan

The Griphan Online School — University Library

IMPRINT

This book is published for the Library of the Griphan Online School. Author and publisher of record: **Nji Griphan**. School address on the web: **griphan.48hourrfprescue.com**. Professional mailbox: **info@griphan.48hourrfprescue.com**. © 2026 Nji Griphan. This book was written for the students of this school. Students enrolled at the school may download and keep it for their own study. The library grows shelf by shelf until it carries the whole university.

A note on this book

Computer science is not the study of a machine; it is the study of information and the processes that transform it. This book builds the foundations — how computers represent and process information, how programs are written and reasoned about, and how data is organised — in plain language, with examples a beginning university

student can follow and practise.

Contents

Chapter 1 — What a Computer Is

Chapter 2 — Representing Information — Bits and Data

Chapter 3 — Algorithms — Recipes for Machines

Chapter 4 — Programming — Telling the Machine

Chapter 5 — Data Structures and Databases

Chapter 6 — Systems, Networks and Staying Safe

Appendix — Worked Examples and Practice

Glossary of Key Terms

Chapter 1

What a Computer Is

A computer is a machine that follows instructions to transform information. This chapter separates the ideas that have survived every change of technology — input, processing, storage, output, and the stored program — from the gadgets that embody them for a decade at a time.

Input, processing, storage, output

Every computing task fits one pattern: information enters (keyboard, sensor, file), is processed according to rules, is stored for keeping, and leaves as output (screen, sound, printed page). Naming the four stages in any system — a phone, a bank's computers, a weather model — is the first analytical habit of computer science.

Hardware and software

Hardware is the physical machinery; software is the instructions it follows. The same hardware runs a word processor today and a game tomorrow because the software changed. This separation — one machine, many possible behaviours — is the computer's defining trick.

The stored program

The stored-program idea keeps instructions in the same memory as data, so a program can be loaded, changed and even written by another program. From this single idea descend compilers, apps, updates — and computer viruses, which are merely programs nobody invited.

Layers of abstraction

No one designs a laptop transistor by transistor. Engineers work in layers — circuits, processors, operating systems, applications — each layer hiding the detail below and offering services above. Abstraction, the disciplined hiding of detail, is the master technique of the whole field.

Key points

- Every task: input, processing, storage, output.
- Software gives one machine many behaviours.
- Abstraction — hiding detail in layers — is the master technique.

Exercises

1. Name the four stages in a mobile-money transaction.
2. Give two examples of abstraction outside computing.
3. Explain the stored-program idea to a friend in three sentences.

Chapter 2

Representing Information — Bits and Data

Inside the machine, everything — numbers, letters, pictures, sound — is represented in binary: patterns of two states. This chapter shows why, and how the representations work well enough to rebuild the world from patterns of nothing but 0 and 1.

Why binary?

Electronic switches are reliably in one of two states, so machines count in base two. A bit is one such choice; a byte, eight bits, holds 256 patterns — enough for a letter, a shade, or a small number. Reliability, not elegance, chose binary; everything else is engineering on top of it.

Numbers in binary

Binary place values double — 1, 2, 4, 8, 16 — so 1011 means 8+0+2+1 = 11. Larger numbers need more bits: 32 bits hold over four billion patterns. Understanding place value in base two demystifies every '32-bit' and '64-bit' label you will ever meet.

Text, images and sound

Text assigns each character a number (Unicode gives every script its codes). Images divide into pixels, each a set of numbers for colour. Sound samples the wave thousands of times per second, each sample a number. Different media, one trick: agree numbers for pieces, then store the numbers.

Compression and error

Files shrink by removing repetition — compression — losslessly where every bit matters (text, programs) and lossily where the eye or ear forgives (photos, music). Stored and transmitted data also gains check digits so errors announce themselves instead of passing silently.

Key points

- All data becomes numbers; all numbers become bits.
- A byte holds 256 patterns; place values double in binary.
- Compression removes repetition; checks catch errors.

Exercises

1. Convert 13 and 27 to binary; convert 11010 back.
2. How many patterns do 16 bits hold?
3. Explain how a photograph becomes numbers, in your own words.

Chapter 3

Algorithms — Recipes for Machines

An algorithm is a precise procedure that solves a class of problems. This chapter teaches you to write, trace and judge algorithms — by correctness first and efficiency second — using searching and sorting, the field's classic teaching examples.

Precision and finiteness

An algorithm must be unambiguous (each step executable without guessing), must finish, and must solve every case of its problem, not merely the examples tried. A recipe that works 'usually' is a draft; the discipline is in the awkward cases — the empty list, the single item, the tie.

Searching

Linear search checks items one by one: simple, and slow on long lists. Binary search halves a sorted list repeatedly — find a name in a thousand-page directory in about ten halvings. Its price is the precondition: the list must be sorted, a lesson that preconditions are part of an algorithm, not fine print.

Sorting

Sorting rearranges data into order. Simple sorts compare and swap neighbours; efficient sorts divide and conquer. Sorting matters beyond tidiness: sorted data enables binary search, duplicate-finding, ranking and merging — half of practical data work begins by sorting something.

Efficiency — counting steps

Algorithms are compared by how their step count grows with input size: doubling input doubles linear search's work but adds one step to binary search's. This growth thinking, not stopwatch timings on one machine, is how computer scientists predict whether a method survives real-world scale.

Key points

- An algorithm must be precise, finite and correct on awkward cases.
- Binary search needs sorted data; preconditions are part of the method.
- Judge algorithms by how work grows with input size.

Exercises

1. Trace binary search for 42 in [3, 9, 15, 21, 28, 35, 42, 50].

2. How many steps does linear search take, worst case, on 1,000,000 items? Binary search?

3. Write an algorithm (in words) to find the largest of a list of numbers.

Chapter 4

Programming — Telling the Machine

Programming translates algorithms into a language a machine can execute. This chapter introduces the universal building blocks — variables, sequence, selection, repetition and functions — that every language shares beneath its surface spelling.

Variables and assignment

A variable is a named store whose value can change: score = 0 names a box and puts 0 in it. Assignment is not equality — score = score + 1 means 'take the old value, add one, store it back'. Most beginner confusion dies once this single distinction is clear.

Sequence, selection, repetition

Programs run statements in sequence; selection (if … then … else) chooses paths by conditions; repetition (loops) repeats work while a condition holds. Every program ever written, from a calculator to a flight controller, is these three structures woven together — the rest is vocabulary.

Functions — naming a procedure

A function packages a procedure under a name, with inputs and an output, so it can be used without re-thinking its insides: abstraction again, at program scale. Libraries of functions are how programmers stand on each other's work instead of starting from the bare machine.

Errors and debugging

Programs fail as syntax errors (the language is broken), runtime errors (the machine is asked the impossible) and logic errors (the program runs and answers wrongly). Debugging is detective work: reproduce the failure, shrink it, and check assumptions one at a time. The bug is always in what you were sure of.

Key points

- Assignment stores; it does not equate.
- Sequence, selection, repetition build every program.
- Debug by reproducing, shrinking and testing assumptions.

Exercises

1. Trace: x = 3; x = x * 2; x = x + 1. What is x?

2. Write pseudocode to print the even numbers from 2 to 20.

3. Classify three program failures as syntax, runtime or logic errors.

Chapter 5

Data Structures and Databases

How data is organised decides what questions are cheap to answer. This chapter surveys the classic structures — arrays, lists, stacks, queues, trees and tables — and the database idea that organises the world's records.

Arrays and lists

An array stores items in numbered positions with instant access by index, at the price of a fixed size. A list chains items so insertions are easy, at the price of walking to reach a position. Choosing between them — access speed versus flexibility — is the archetypal data-structure trade-off.

Stacks, queues and trees

A stack serves last-in-first-out (the undo button's memory); a queue serves first-in-first-out (the printer's fairness). Trees branch — folders inside folders, family trees, decision trees — and underlie everything from file systems to the indexes that make database searches fast.

Tables and keys

A database table stores records as rows described by columns, with a key identifying each row uniquely. Related tables link by keys — a student's key appears in the enrolment table — so each fact is stored once and combined by queries when needed.

Queries — asking precisely

A query states exactly which records and fields are wanted: all students in a class, sorted by name. Structured query languages make such questions precise and repeatable. The researcher's tidy table of Volume I has grown up into the database; the discipline is the same.

Key points

- Structure decides what is cheap to ask.
- Stacks: last in, first out. Queues: first in, first out.
- Keys identify rows and link tables so facts are stored once.

Exercises

1. Choose structures: a printer's jobs; an undo history; a school's student records. Justify each.

2. Design a students table: columns and its key.

3. Write, in words, a query for all students in one class sorted by surname.

Chapter 6

Systems, Networks and Staying Safe

Computers earn their keep connected — to users through operating systems and to each other through networks. This final chapter explains operating systems, the internet's working ideas, and the security habits every student of computing must practise.

The operating system

The operating system manages the machine's resources — processor time, memory, files, devices — and offers them to applications through orderly doors. It is traffic police, librarian and translator in one: without it, every program would have to drive the hardware itself.

Networks and the internet

Networks link machines so they can exchange messages by agreed rules (protocols). The internet is a network of networks using common protocols, addressing each machine and breaking messages into packets that find their own way and are reassembled on arrival. The web is one service riding on it, not the internet itself.

Security — the human layer

Most breaches enter through people: reused passwords, phishing messages, unpatched software. Defence is layered — strong unique passwords, a second factor where offered, updates applied, backups kept, and suspicion toward urgency in messages. No software repairs a password given away.

Your road in computing

Foundations first — this book — then a language practised daily, small projects finished rather than grand ones started, and mathematics alongside. Computing rewards those who build understanding layer by layer, exactly as its machines are built.

Key points

- The operating system manages resources for all programs.
- The internet: common protocols, addresses, packets. The web rides on it.
- Security is layered, and the human layer decides most outcomes.

Exercises

1. List three jobs your phone's operating system does while you read this.

2. Explain packets to a friend using the postal system.

3. Audit your own accounts against this chapter's security list.

Appendix — Worked Examples and Practice

From Chapter 1 — What a Computer Is

Worked example: sending a message — input: your typed words and the contact you chose; processing: the app formats and encrypts them and finds the route; storage: the message rests on servers and both phones; output: words appear on your friend's screen. Four stages, one pattern — the same pattern a supercomputer follows, only faster.

From Chapter 2 — Representing Information — Bits and Data

Worked example: 27 in binary — the largest power of two not exceeding 27 is 16 (remainder 11), then 8 (remainder 3), then 2 (remainder 1), then 1. So 27 = 16+8+2+1 = 11011. Check by place values from the right: 1, 1, 0, 1, 1 against 1, 2, 4, 8, 16: 1+2+0+8+16 = 27. The method and the check are both the lesson.

From Chapter 3 — Algorithms — Recipes for Machines

Worked example: searching 1,000,000 sorted records — linear search may take 1,000,000 steps; binary search halves: 500,000, 250,000 … after about 20 halvings one candidate remains (2^20 ≈ 1,000,000). Twenty steps against a million is why the world's directories, indexes and databases are built on sorted order.

From Chapter 4 — Programming — Telling the Machine

Worked example: x = 3 stores 3. x = x * 2 takes the current 3, doubles it, stores 6. x = x + 1 stores 7. Final answer: 7. Read '=' as 'becomes' and every trace becomes this simple — the machine reads the right side with old values, then writes the left side once.

From Chapter 5 — Data Structures and Databases

Worked example: a library stored as one giant list must walk every record to find one book — the linear search problem again. Add an index tree ordered by title and the search becomes binary: the index is extra storage bought to make questions cheap. Every database design is this bargain, struck deliberately.

From Chapter 6 — Systems, Networks and Staying Safe

Worked example: a message 'Your account will close in one hour — verify your password here' carries the three classic phishing marks: manufactured urgency, a threat, and a link asking for credentials. The defence costs nothing: stop, reach the service by its own app or address — never by the message's link — and check there. Ninety per cent of security is this pause.

Further Practice — Review and Discussion

Work through these prompts in writing; they are the revision spine of this book. Answer in full sentences, with examples from your own place of study wherever the prompt allows.

Chapter 1 — What a Computer Is

- Explain in your own words, with one example: Every task: input, processing, storage, output.
- Explain in your own words, with one example: Software gives one machine many behaviours.
- Explain in your own words, with one example: Abstraction — hiding detail in layers — is the master technique.

Discussion: where does the subject of this chapter — what a computer is — touch your own community, farm, school or workplace? Write one page, naming real places and real constraints, and end with one question you would research next.

Chapter 2 — Representing Information — Bits and Data

- Explain in your own words, with one example: All data becomes numbers; all numbers become bits.
- Explain in your own words, with one example: A byte holds 256 patterns; place values double in binary.
- Explain in your own words, with one example: Compression removes repetition; checks catch errors.

Discussion: where does the subject of this chapter — representing information — bits and data — touch your own community, farm, school or workplace? Write one page, naming real places and real constraints, and end with one question you would research next.

Chapter 3 — Algorithms — Recipes for Machines

- Explain in your own words, with one example: An algorithm must be precise, finite and correct on awkward cases.
- Explain in your own words, with one example: Binary search needs sorted data; preconditions are part of the method.
- Explain in your own words, with one example: Judge algorithms by how work grows with input size.

Discussion: where does the subject of this chapter — algorithms — recipes for machines — touch your own community, farm, school or workplace? Write one page, naming real places and real constraints, and end with one question you would research next.

Chapter 4 — Programming — Telling the Machine

- Explain in your own words, with one example: Assignment stores; it does not equate.
- Explain in your own words, with one example: Sequence, selection, repetition build every program.
- Explain in your own words, with one example: Debug by reproducing, shrinking and testing assumptions.

Discussion: where does the subject of this chapter — programming — telling the machine — touch your own community, farm, school or workplace? Write one page, naming real places and real constraints, and end with one question you would research next.

Chapter 5 — Data Structures and Databases

- Explain in your own words, with one example: Structure decides what is cheap to ask.
- Explain in your own words, with one example: Stacks: last in, first out. Queues: first in, first out.
- Explain in your own words, with one example: Keys identify rows and link tables so facts are stored once.

Discussion: where does the subject of this chapter — data structures and databases — touch your own community, farm, school or workplace? Write one page, naming real places and real constraints, and end with one question you would research next.

Chapter 6 — Systems, Networks and Staying Safe

- Explain in your own words, with one example: The operating system manages resources for all programs.
- Explain in your own words, with one example: The internet: common protocols, addresses, packets. The web rides on it.
- Explain in your own words, with one example: Security is layered, and the human layer decides most outcomes.

Discussion: where does the subject of this chapter — systems, networks and staying safe — touch your own

community, farm, school or workplace? Write one page, naming real places and real constraints, and end with one question you would research next.

Glossary of Key Terms

Algorithm — A precise, finite procedure solving a class of problems.

Binary — Base-two representation using only 0 and 1.

Bit — A single binary choice; eight bits make a byte.

Abstraction — Hiding detail behind a layer that offers services upward.

Function — A named procedure with inputs and an output.

Database — Organised tables of records linked by keys and answered by queries.

Protocol — Agreed rules by which machines exchange messages.

Operating system — The software managing a machine's resources for all other programs.

End of book. Nji Griphan · griphan.48hourrfprescue.com · info@griphan.48hourrfprescue.com