ARLIZ

A JOURNEY THROUGH ARRAYS

Mahdi

In Praise of Arliz

MAHDI

This book evolves. Every insight gainedwhether a circuit, a structure, or a simple ideais absorbed and integrated.

FIRST EDITION

May 30, 2025

To those who build from first principles.
To the silent thinkers who design before they speak.
To the ones who see in systems
not just machines, but metaphors.
This is for you.

Preface

Every book has its own story, and this book is no exception. If I were to summarize the process of creating this book in one word, that word would be improvised. Yet the truth is that Arliz is the result of pure, persistent curiosity that has grown in my mind for years. What you are reading now could be called a technical book, a collection of personal notes, or even a journal of unanswered questions and curiosities. But Ioffi-ciallycall it a *book*, because it is written not only for others but for myself, as a record of my learning journey and an effort to understand more precisely the concepts that once seemed obscure and, at times, frustrating.

The story of Arliz began with a simple feeling: **curiosity**. Curiosity about what an array truly is. Perhaps for many this question seems trivial, but for me this worden-countered again and again in algorithm and data structure discussionsalways raised a persistent question.

Every time I saw terms like `array`, `stack`, `queue`, `linked list`, `hash table`, or `heap`, I not only felt confused but sensed that something fundamental was missing. It was as if a key piece of the puzzle had been left out. The first brief, straightforward explanations I found in various sources never sufficed; they assumed you already knew exactly what an array is and why you should use it. But I was looking for the *roots*. I wanted to understand from zero what an array means, how it was born, and what hidden capacities it holds.

That realization led me to decide: *If I truly want to understand, I must start from zero.*

There is no deeper story behind the name Arliz. There is no hidden philosophy or special inspirationjust a random choice. I simply declared: *This book is called Arliz.* You may pronounce it "Ar-liz," "Array-Liz," or any way you like. I personally say "ar-liz." That is allsimple and arbitrary.

But Arliz is not merely a technical book on data structures. In fact, **Arliz grows along-side me**.

Whenever I learn something I deem worth writing, I add it to this book. Whenever I feel a section could be explained better or more precisely, I revise it. Whenever a new idea strikes mean algorithm, an exercise, or even a simple diagram to clarify a struc-

tureI incorporate it into Arliz.
This means Arliz is a living project. As long as I keep learning, Arliz will remain alive.
The structure of this book has evolved around a simple belief: true understanding begins with context. Thats why Arliz doesnt start with code or syntax, but with the origins of computation itself. We begin with the earliest tools and ideascounting stones, the abacus, mechanical gears, and early notions of logiclong before transistors or binary digits came into play. From there, we follow the evolution of computing: from ancient methods of calculation to vacuum tubes and silicon chips, from Babbages Analytical Engine to the modern microprocessor. Along this journey, we discover that concepts like arrays arent recent inventionsthey are the culmination of centuries of thought about how to structure, store, and process information.
In writing this book, I have always tried to follow three principles:

- **Simplicity of Expression:** I strive to present concepts in the simplest form possible, so they are accessible to beginners and not superficial or tedious for experienced readers.

- **Concept Visualization:** I use diagrams, figures, and visual examples to explain ideas that are hard to imagine, because I believe visual understanding has great staying power.

- **Clear Code and Pseudocode:** Nearly every topic is accompanied by code that can be easily translated into major languages like C++, Java, or C#, aiming for both clarity and practicality.

An important note: many of the algorithms in Arliz are implemented by myself. I did not copy them from elsewhere, nor are they necessarily the most optimized versions. My goal has been to understand and build them from scratch rather than memorize ready-made solutions. Therefore, some may run slower than standard implementationsor sometimes even faster. For me, the process of understanding and constructing has been more important than simply reaching the fastest result.
Finally, let me tell you a bit about myself: I am **Mehdi**. If you prefer, you can call me by my alias: *Genix*. I am a student of Computer Engineering (at least at the time of writing this). I grew up with computersfrom simple games to typing commands in the terminaland I have always wondered what lies behind this screen of black and green text. There is not much you need to know about me, just that I am someone who works with computers, sometimes gives them commands, and sometimes learns from them.
I hope this book will be useful for understanding concepts, beginning your learning

journey, or diving deeper into data structures.
Arliz is freely available. You can access the PDF, LaTeX source, and related code at:

https://github.com/m-mdy-m/Arliz

In each chapter, I have included exercises and projects to aid your understanding. Please do not move on until you have completed these exercises, because true learning happens only by solving problems.
I hope this book serves you wellwhether for starting out, reviewing, or simply satisfying your curiosity. And if you learn something, find an error, or have a suggestion, please let me know. As I said: *This book grows with me.*

Acknowledgments

I would like to express my gratitude to everyone who supported me during the creation of this book. Special thanks to the open-source community for their invaluable resources and to all those who reviewed early drafts and provided feedback.

Contents

How to Read This Book

This book is not like most technical books you've probably encountered. It doesn't start with "Here's how to declare an array" or jump straight into syntax and algorithms. Instead, Arliz takes you on a journeya long, winding path that begins thousands of years ago with humans counting on their fingers and ends with the sophisticated data structures we use today.

I know what you're thinking: "Why should I care about ancient history when I just want to learn arrays?" That's a fair question, and I've asked myself the same thing many times while writing this book. Here's the thingunderstanding where something comes from changes how you think about it. When you know that arrays are not just programming constructs but the culmination of humanity's age-old quest to organize information, you start to see them differently. You begin to understand not just *how* they work, but *why* they work the way they do.

Arliz is structured in seven parts, each building upon the previous one:

Part 1: Philosophical & Historical Foundations is where we are now. This part traces the human journey from basic counting to systematic representation. We explore ancient civilizations, their counting systems, the invention of the abacus, and the gradual development of mathematical thinking that made modern computation possible. This isn't just history for history's sakeit's the conceptual foundation that makes everything else make sense.

Part 2: Mathematical Fundamentals dives into the mathematical concepts that underlie all data structures. We cover set theory, functions, mathematical logic, and discrete mathematics. If Part 1 gives you the historical context, Part 2 gives you the mathematical tools to understand why data structures work the way they do.

Part 3: Data Representation explores how information is encoded in digital systems. We look at number systems, binary representation, character encoding, and the various ways computers store and manipulate data. This is where the abstract concepts from Parts 1 and 2 start to become concrete.

Part 4: Computer Architecture & Logic examines the hardware foundations of computation. We explore logic gates, processor architecture, memory systems, and how the physical structure of computers influences the way we organize data.

Part 5: Array Odyssey is the heart of the book. Here, we finally meet arrays in all their glorynot as mysterious programming constructs, but as the natural evolution of thousands of years of human thought about organizing information. We explore their implementation, behavior, and applications in depth.

Part 6: Data Structures & Algorithms expands beyond arrays to explore the broader landscape of data structures. Having understood arrays thoroughly, we can now appreciate how other structures like linked lists, trees, and graphs relate to and build upon array concepts.

Part 7: Parallelism & Systems looks at how data structures behave in complex, multi-threaded, and distributed systems. This is where we explore the cutting edge of modern computation.

Now, you might be wondering: "Do I really need to read all of this? Can't I just skip to the arrays part?"

The honest answer is: it depends on who you are and what you want to get out of this book.

If you're a complete beginnersomeone who's never programmed before, or who's just starting to learn about computer sciencethen yes, I strongly recommend reading the book from beginning to end. The concepts build upon each other in a way that's designed to create a solid, unshakeable foundation for your understanding.

If you're an experienced programmer who just wants to deepen your understanding of arrays specifically, you could potentially start with Part 5. However, I'd encourage you to at least skim Parts 1 and 2. You might be surprised by how much the historical and mathematical context enriches your understanding of concepts you thought you already knew.

If you're somewhere in betweenmaybe you know some programming but feel like you're missing fundamental conceptsthen Parts 2, 3, and 4 might be your sweet spot. You can always come back to Part 1 later when you want to understand the bigger picture.

For students and educators, each part serves a different pedagogical purpose. Part 1 provides historical context and motivation. Parts 2-4 build theoretical foundations. Parts 5-7 provide practical application and advanced concepts. You can use different parts for different courses or learning objectives.

But here's what I really want you to understand: this book is not just about consuming information. It's about building intuition. Each part includes exercises, thought experiments, and projects designed to help you internalize the concepts. Don't skip these. They're not just busy workthey're carefully designed to help you develop the kind of deep, intuitive understanding that will serve you throughout your career.

One more thing: as I mentioned in the preface, this book grows with me. If you find errors, have suggestions, or discover better ways to explain something, please let me know. This is a living document, and your feedback helps make it better for everyone.

So, whether you're here for the full journey or just part of it, welcome to Arliz. Let's explore the fascinating world of arrays togetherstarting from the very beginning.

Part I

Philosophical & Historical Foundations

Introduction

Long before arrays existed as data structures in programming languageslong before computers, algorithms, or even formal mathematicshumans possessed an innate drive to organize, count, and systematically represent the world around them. This part of our journey explores not just the technical evolution of computational tools, but the profound intellectual transformation of human thought about order, sequence, and structured information.

Arrays are not merely programming constructs. They are the culmination of humanity's oldest and most fundamental intellectual pursuit: the systematic organization of information. Their conceptual roots stretch back thousands of years, embedded in the clay tablets of Mesopotamia, the geometric patterns of ancient Egypt, the bead arrangements of the abacus, and the philosophical frameworks of classical mathematics. To truly understand arrays, we must first understand the human mind's relentless quest to impose order upon chaos, to find patterns within complexity, and to create systems that can capture, manipulate, and transform structured knowledge.

Our exploration begins in the prehistoric dawn of human consciousness, when our ancestors first felt compelled to count beyond their fingers, to track seasons and harvests, to record transactions and astronomical observations. We witness the birth of positional notation in ancient Mesopotamiathe revolutionary idea that the **position** of a symbol could carry meaning, laying the conceptual groundwork for array indexing. We follow the development of the abacus across civilizations, seeing how different cultures refined this early computational array, creating sophisticated systems for parallel calculation that echo modern array operations.

As we progress through classical antiquity, we encounter the Greek philosophers who first formalized concepts of **sets**, **sequences**, and **ordered arrangements**. Aristotle's categorical thinking, Euclid's systematic geometry, and the Pythagorean exploration of number patterns all contributed essential building blocks for understanding structured data. The Chinese mathematical tradition, with its matrix-like arrangements for solving systems of equations, demonstrates early intuitive grasp of multidimensional data organization.

The medieval period brings us algorithmic thinkingAl-Khwarizmi's systematic procedures, the revolutionary introduction of zero and positional notation from the Hindu-Arabic tradition, and the monastic scriptoriums that pioneered systematic knowledge organization. These developments mark the transition from intuitive arrangement to formal, reproducible methods of data manipulation.

The Renaissance and early modern period witness the birth of symbolic thinkingViète's algebraic notation, Descartes' coordinate systems, Pascal's triangular arrangements of combinatorial coefficients. Each breakthrough represents a step toward the abstract, systematic representation that enables modern computational thinking. By the time we reach the threshold of mechanical computation with Pascal's calculator and Leibniz's universal symbolic aspirations, the conceptual foundations for array-based thinking are fully established.

This historical foundation is not mere academic curiosity. Every concept explored in later parts of this bookfrom basic array operations to complex algorithmic optimizationsbuilds upon intellectual frameworks developed across millennia. Understanding this deep history provides not just context, but genuine insight into why arrays work the way they do, why certain operations are natural while others are complex, and how the fundamental patterns of structured thinking manifest in modern computational systems.

When you encounter array indexing, you're participating in a tradition that began with Mesopotamian scribes arranging cuneiform symbols on clay tablets. When you manipulate matrices, you're extending methods pioneered by Chinese mathematicians over two thousand years ago. When you design data structures, you're continuing humanity's ancient quest to create order from complexity, to find systematic methods for representing and transforming information.

This part prepares you for the mathematical formalism of Part 2, the technical implementation details of later sections, and ultimately, for a deeper appreciation of arrays as both practical tools and profound expressions of human intellectual achievement.

How to Read

This part is structured as a chronological journey through humanity's development of systematic thinking about information organization. Each chapter builds upon previous concepts while introducing new layers of complexity. The progression is intentionally gradualfrom concrete counting methods to abstract mathematical frameworksmirroring how human understanding evolved over millennia.

For the Complete Journey: Read chapters sequentially. This provides the full historical and conceptual foundation, showing how each civilization and era contributed essential elements to our modern understanding of structured data. Pay particular attention to recurring themes: position and place-value systems, systematic arrangement methods, symbolic representation, and the gradual abstraction from concrete tools to mathematical concepts.

For Focused Study: If you're primarily interested in specific aspects, you can emphasize certain chapters while skimming others.

Connecting to Later Parts: As you read, note how concepts introduced here reappear in mathematical formalization (Part 2), data representation (Part 3), and implementation details (Parts 4-7). The philosophical frameworks developed in early chapters provide context for technical decisions made in modern computing systems.

Each chapter includes timeline markers and focuses on specific conceptual developments. Don't merely read for historical factsengage with the underlying ideas. Ask yourself: How did this development change how humans thought about organized information? What limitations did it overcome? What new possibilities did it create? This active engagement will deepen your understanding of both historical development and modern applications.

Chapter 1

The Primordial Urge to Count and Order

1.1 The Philosophy of Measurement and Representation

1.2 From Stones to Symbols: Early Counting Systems

1.3 The Birth of Position and Place

1.4 Sacred Geometry and Ordered Patterns

Chapter 2

Ancient Civilizations and Systematic Thinking

2.1 Mesopotamian Clay Tablets and Cuneiform Arrays

2.2 Egyptian Hieroglyphic Systems and Papyrus Records

2.3 Chinese Oracle Bones and Early Matrix Concepts

2.4 Indus Valley: Lost Systems of Organization

Chapter 3

The Abacus Revolution: Humanity's First Computational Array

3.1 Mesopotamian Counting Boards: The Genesis

3.2 Chinese Suanpan: Perfecting the Art

3.3 Greco-Roman Abacus: Spreading the Concept

3.4 Philosophical Implications: State, Position, and Transformation

Chapter 4

Classical Mathematical Foundations

4.1 Greek Philosophy of Numbers and Sets

4.2 Aristotelian Logic and Categorical Thinking

4.3 Diophantine Equations and Early Matrix Operations

4.4 Chinese Mathematical Texts: The Nine Chapters

Chapter 5

Medieval Synthesis and the Algorithmic Awakening

5.1 Islamic Golden Age: Al-Khwarizmi and Systematic Procedures

5.2 The Hindu-Arabic Numeral Revolution

5.3 Monastic Scriptoriums: Knowledge Preservation and Organization

5.4 Persian and Central Asian Mathematical Traditions

Chapter 6

High Medieval Mathematical Renaissance

6.1 European University System: Systematic Mathematical Education

6.2 Fibonacci and the Liber Abaci: Bringing Systems to Europe

6.3 Islamic Mathematical Synthesis: Al-Samaw'al and Polynomial Arrays

6.4 Medieval Astronomy: Tabular Data and Systematic Observation

Chapter 7

Late Medieval to Renaissance Transition

7.1 Scholastic Method: Systematic Logical Analysis

7.2 Commercial Mathematics: Practical Systematic Calculations

7.3 Medieval Islamic Algebraic Traditions

7.4 Early Renaissance Humanism: Rediscovery and Systematization

Chapter 8

Renaissance Symbolic Revolution

8.1 Viète's Algebraic Symbolism: The Birth of Abstract Representation

8.2 Cardano and Systematic Solution Methods

8.3 Decimal System Standardization: Stevin and Systematic Notation

8.4 Renaissance Art and Mathematical Perspective: Systematic Spatial Representation

Chapter 9

Early Modern Mathematical Systematization

9.1 Cartesian Coordinate System: Systematic Spatial Representation

9.2 Leibniz's Characteristica Universalis: Dreams of Universal Symbolism

9.3 Pascal's Triangle and Combinatorial Arrays

9.4 Early Probability Theory: Systematic Analysis of Uncertainty

Chapter 10

The Threshold of Mechanical Computation

10.1 Pascal's Calculator: Mechanizing Systematic Calculation

10.2 Leibniz's Step Reckoner: Toward Universal Calculation

10.3 Euler's Systematic Mathematical Notation

10.4 The Encyclopédie: Systematic Organization of Human Knowledge

Part II

Mathematical Fundamentals

Introduction

How to Read

Part III

Data Representation

Introduction

How to Read

Part IV

Computer Architecture & Logic

Introduction

How to Read

Part V

Array Odyssey

Introduction

How to Read

Part VI

Data Structures & Algorithms

Introduction

How to Read

Part VII

Parallelism & Systems

Introduction

How to Read

Part VIII

Synthesis & Frontiers

Introduction

How to Read