ARLIZ

A JOURNEY THROUGH ARRAYS

Mahdi

ARLIZ

Arliz

In Praise of

This book evolvesevery insight gained, whether a circuit,
a structure, or a simple idea, is absorbed into its living form.

First Edition

June 8, 2025

ARLIZ: A Living Architecture of Computing

First Edition

Repository and Source. The full source (LaTeX, images, and accompanying code) is available at
https://github.com/m-mdy-m/Arliz
under the same CC BY-SA 4.0 license.

This book is typeset using LaTeX with the Charter and Palatino font families, microtype enhancements, and TikZ illustrations for diagrams. All figures, diagrams, and code examples are either authored by Mahdi Genix or used with permission under permissive licenses.

Cover and Design Credits:
Cover design by Mahdi and Studio Montage-inspired elements. Title typography arranged with TikZ overlays to reflect the living, evolving nature of Arliz.

Development Team:

Author	Mahdi
Repository Maintenance	Mahdi
Editorial Guidance	(Open-source community contributions)
LaTeX Template and Styling	Mahdi
Illustrations and Diagrams	Mahdi and contributors

Library of Congress Control Number: 2025-ARLIZ
ISBN: Pending (Digital version only)
First Printing: 2025
Printed in Iran (when available), or distributed digitally worldwide.
The text of this book is printed on acid-free, archival-quality paper.
URLs, repository links, and license details were last updated on June 8, 2025.

Contents

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 Iofficiallycall 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 wordencountered 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 was no deeper story behind the name Arliz at firstjust a random choice. But over time, I found a fitting expansion:

$$\text{Arliz} = \text{Arrays, Reasoning, Logic, Identity, Zero}$$

This backronym captures the essence of the book:

- Arrays: The fundamental data structure we aim to explore from its origins.
- Reasoning: The logical thinking behind data organization.
- Logic: The reasoning and thought processes behind how computers organize and manipulate data.
- Identity: The notion of distinguishing, indexing, and giving identity to elements within structures.
- Zero: The philosophical and mathematical concept of nothing from which all computation, counting, and indexing originate.

In other words, Arliz is not merely a random stringit signifies the core pillars that guide this journey: from the first zero to the very way we reason about data. You may pronounce it Ar-liz, Array-Liz, or however you like. I personally say ar-liz.

So yes, my naming process goes like this: pick a random name and then look for a good backronym to justify it. Very scientific, I know!

But Arliz is not merely a technical book on data structures. In fact, Arliz grows alongside 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 structureI 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 Mahdi. 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.

How to Read This Book

Look, I get it. You picked up a book called "Arliz" expecting to learn about arrays, and here I am starting with ancient civilizations and counting stones. You're probably thinking, "What the hell does Mesopotamian clay tablets have to do with int[] myArray = new int[10]?" And honestly? That's a perfectly reasonable question. If you think this approach is ridiculous, you're welcome to close this PDF right now. Or if you have the physical book, feel free to use it as a makeshift heating deviceit's thick enough to provide decent warmth.
But before you do that, let me make my case.

Why This Book Exists (And Why You Might Actually Want to Read It)

Every programming book I've ever read starts the same way: "Here's an array. It stores elements. Here's how you declare one. Moving on." And you know what? That approach produces programmers who can use arrays but don't truly understand them. They can write code that works, but when things breakand they will breakthey're lost. They treat arrays like black magic: mysterious entities that sometimes work and sometimes don't, for reasons that remain forever opaque.
This book exists because I refuse to accept that level of understanding. When I started programming, I wasn't satisfied with "arrays are containers for data." I wanted to know why they exist, how they really work, and what makes them tick at the deepest level. The more I dug, the more I realized that understanding arraystruly understanding themrequires understanding the entire intellectual history that led to their creation.
Here's the thing: arrays aren't just programming constructs. They're the evolutionary culmination of humanity's oldest intellectual pursuitthe systematic organization of information. Every time you write arr[i], you're participating in a tradition that stretches back to ancient Mesopotamian scribes who first realized that the position of a symbol could carry meaning. When you manipulate multidimensional arrays, you're using mathematical concepts that Chinese mathematicians developed over two thousand years ago. When you optimize array operations, you're applying algorithmic thinking that emerged from Islamic mathematical traditions.
Understanding this history doesn't just give you contextit gives you intuition. When you know why arrays work the way they do, you can predict their behavior. When you understand the mathematical principles underlying their structure, you can optimize their usage. When you grasp the conceptual frameworks that enabled their creation, you can extend and adapt them in ways that would be impossible otherwise.
But more than that, this historical perspective changes how you think about programming itself. Instead of seeing yourself as someone who memorizes syntax and follows

patterns, you start to see yourself as part of a continuous intellectual tradition. You're not just using toolsyou're participating in humanity's ongoing quest to create order from chaos, to build systems that can capture, manipulate, and transform structured knowledge.

What You're Getting Into

This book is structured as a journeynot just through the technical aspects of arrays, but through the entire conceptual landscape that makes arrays possible. It's organized into seven parts, each building upon the previous one:

Part 1: Philosophical & Historical Foundations

Yes, we start with ancient history. No, this isn't academic masturbation. We trace the human journey from basic counting to systematic representation, exploring how different civilizations developed the conceptual tools that make modern computation possible. We look at the invention of positional notation, the development of the abacus, the emergence of algorithmic thinking, and the philosophical frameworks that enabled abstract mathematical representation.

Why does this matter? Because every array operation you'll ever perform builds on concepts developed in this part. Array indexing is a direct descendant of positional notation. Multidimensional arrays extend geometric thinking developed by ancient mathematicians. Algorithmic optimization applies systematic procedures that emerged from medieval Islamic mathematics.

Part 2: Mathematical Fundamentals

Here we transform historical intuition into precise mathematical language. We develop set theory, explore functions and relations, dive into discrete mathematics, and build the linear algebra foundations that directly enable array operations. This isn't abstract theoryit's the mathematical machinery that makes arrays work.

If you skip this part, you'll forever be mystified by why certain array operations are efficient while others are expensive, why some algorithms work better with particular data arrangements, and how to reason about the mathematical properties of the code you write.

Part 3: Data Representation

We explore how information is encoded in digital systemsnumber systems, binary representation, character encoding, and the various ways computers store and manipulate data. This is where the abstract concepts from the first two parts become concrete.

Understanding data representation is crucial for working with arrays because it determines how array elements are stored, how memory is allocated, and how operations are performed at the hardware level.

Part 4: Computer Architecture & Logic

We examine the hardware foundations of computationlogic gates, processor architecture, memory systems, and how the physical structure of computers influences the way we organize data. This part connects software concepts to hardware realities.

Arrays don't exist in a vacuum. They're implemented on real hardware with specific characteristics and limitations. Understanding this hardware foundation is essential for writing efficient array-based code.

Part 5: Array Odyssey

Finally, we meet arrays in all their glory. But by this point, they won't be mysterious constructsthey'll be the natural evolution of thousands of years of human thought about organizing information. We explore their implementation, behavior, and applications in unprecedented depth.

This is where everything comes together. The historical foundations provide context, the mathematical frameworks provide analytical tools, the representation and architecture parts provide implementation understanding, and now we can explore arrays as sophisticated, well-understood mathematical objects.

Part 6: Data Structures & Algorithms

Having understood arrays thoroughly, we expand to explore the broader landscape of data structures. We see how other structures like linked lists, trees, and graphs relate to and build upon array concepts.

This part shows how the deep understanding of arrays you've developed transfers to other data structures and enables more sophisticated algorithmic thinking.

Part 7: Parallelism & Systems

We look at how data structures behave in complex, multi-threaded, and distributed systems. This is where we explore the cutting edge of modern computation and see how classical array concepts extend to contemporary challenges.

How to Actually Read This Book

Now for the practical question: Do you really need to read all of this? The answer depends on who you are and what you want to achieve.

If you're a complete beginner: Yes, read everything from start to finish. The concepts build systematically, and skipping parts will leave gaps in your understanding that will haunt you later. This book is designed to take you from zero knowledge to deep, intuitive understanding.

If you're an experienced programmer who wants to deepen your array knowledge: You could potentially start with Part 5, but I strongly recommend at least skimming Parts 1 and 2. You'll be surprised how much the historical and mathematical context enriches concepts you thought you already understood. Parts 3 and 4 will fill in hardware and representation details that most programmers never learn properly.

If you're somewhere in between: Parts 2, 3, and 4 might be your sweet spot. You can always circle back to Part 1 when you want the bigger picture, and jump ahead to Part 5 when you're ready for the main event.

If you're a student or educator: Different parts serve different pedagogical purposes. Part 1 provides motivation and historical context. Parts 2-4 build theoretical foundations. Parts 5-7 provide practical application and advanced concepts. Use whatever combination serves your learning objectives.

But here's what I really want you to understand: this isn't a reference manual. It's not designed for you to flip to specific sections when you need to remember syntax. This is a book about building deep, intuitive understandingthe kind of understanding that transforms how you think about programming and data structures.

Each part includes exercises, thought experiments, and projects. Don't skip these. They're not busy workthey're carefully designed to help you internalize concepts and develop the kind of mathematical intuition that separates good programmers from great ones.

A Warning About Expectations

This book grows with me. It's a living document that evolves as I learn and discover better ways to explain concepts. If this bothers youif you want a static, finished productthen this probably isn't the book for you. But if you're excited by the idea of participating in an ongoing exploration of fundamental concepts, then welcome aboard.

You'll find errors. You'll discover sections that could be clearer. You'll think of better examples or more intuitive explanations. When that happens, let me know. This book improves through community engagement, and your feedback makes it better for everyone.

Also, don't expect this to be a quick read. Building deep understanding takes time. The historical and mathematical foundations require patience and sustained attention. The later technical sections demand careful study and practical application. This isn't a book you read on a weekendit's a book you work through over months, returning to sections as your understanding deepens.

Why This Matters

At the end of the day, this book exists because I believe programmers deserve better than shallow, cookbook-style education. You deserve to understand not just how to use arrays, but why they work, where they came from, and what they represent in the broader context of human intellectual achievement.

When you finish this book, you won't just know how to declare and manipulate arrays. You'll understand them as mathematical objects with precise properties and behaviors. You'll be able to predict their performance characteristics, optimize their usage, and extend their applications in ways that weren't possible before. You'll see connections between arrays and other areas of mathematics and computer science that will inform your thinking for years to come.

More importantly, you'll have developed a way of thinking about programming that goes beyond memorizing syntax and following patterns. You'll understand the deep principles that make computation possible, and you'll be able to apply those principles to solve problems that don't have cookbook solutions.

So if you're ready for that journeyif you're willing to invest the time and mental energy required to build genuine understandingthen let's begin. We're going to start with humans counting on their fingers, and we're going to end up with sophisticated data structures that can process information in ways that would seem magical to our ancestors.

And if you still think starting with ancient history is ridiculous? Well, you can always use this book as a heating device. Just make sure to recycle it responsibly when you're done.

Welcome to Arliz. Let's explore the fascinating world of arrays together—from the very beginning.

Part I

Philosophical & Historical Foundations

Introduction

Before we dive into syntax and algorithms, we need to understand something fundamental: every time you create an array, you're participating in a tradition that stretches back thousands of years. When ancient Mesopotamians arranged symbols on clay tablets, when Chinese mathematicians organized numbers in grid patterns, when Islamic scholars developed systematic proceduresthey were all working toward the same goal that drives modern programming: turning chaos into order through structured thinking.

This part traces that journey from the first human attempts at counting to the threshold of mechanical computation. We'll see how the abacus anticipated array operations, how positional notation laid the groundwork for indexing, and how mathematical philosophy shaped the way we think about organized data.

Why start here? Because understanding the why behind arrays changes everything. Instead of memorizing rules, you'll develop intuition. Instead of fighting with concepts, you'll see their natural logic. When you know that arrays are humanity's answer to an age-old problem, they stop being mysterious programming constructs and become what they really are: elegant solutions to the fundamental challenge of organizing information.

Chapter 1

The Primordial Urge to Count and Order

1.1 The Philosophy of Measurement and Human Consciousness

1.2 Paleolithic Counting: Bones, Stones, and Fingers

1.3 Neolithic Revolution: Agriculture and the Need for Records

1.4 Proto-Writing and Symbolic Representation

Chapter 2

Mesopotamian Foundations of Systematic Thinking

Chapter 3

Egyptian Systematic Knowledge and Geometric Arrays

3.1 Hieroglyphic Number Systems and Decimal Thinking

3.2 The Rhind Papyrus: Systematic Mathematical Methods

3.3 Sacred Geometry and Architectural Arrays

3.4 Egyptian Fractions and Systematic Decomposition

Chapter 4

Indus Valley Civilization: Lost Systems of Order

Chapter 5

Ancient Chinese Mathematical Matrices and Systematic Thinking

Chapter 6

The Abacus Revolution Across Civilizations

Chapter 7

Greek Mathematical Philosophy and Logical Foundations

7.1 Pythagorean Number Theory and Systematic Patterns

7.2 Euclidean Geometry: The Axiomatic Method

7.3 Aristotelian Categories: The Logic of Classification

7.4 Platonic Mathematical Idealism

Chapter 8

Hellenistic Mathematical Innovations

8.1 Alexandrian Mathematical Synthesis

8.2 Apollonius and Systematic Geometric Investigation

8.3 Diophantine Analysis and Early Algebraic Thinking

8.4 Greek Mechanical Devices and Computational Aids

Chapter 9

Indian Mathematical Breakthroughs

9.1 The Revolutionary Concept of Zero

9.2 Hindu-Arabic Numerals and Place-Value Revolution

9.3 Aryabhata and Early Algorithmic Thinking

9.4 Indian Combinatorics and Systematic Enumeration

Chapter 10

The Islamic Golden Age and Algorithmic Revolution

10.1 Al-Khwarizmi: The Birth of Algebra and Algorithms

10.2 House of Wisdom: Systematic Knowledge Preservation

10.3 Persian and Arab Mathematical Innovations

10.4 Islamic Geometric Patterns and Systematic Design

Chapter 11

Medieval European Synthesis and University System

Chapter 12

Late Medieval Innovations and Mechanical Aids

Chapter 13

Renaissance Symbolic Revolution

13.1 Viète's Algebraic Symbolism: Abstract Mathematical Representation

13.2 Cardano and Systematic Classification of Solution Methods

13.3 Stevin and Decimal System Standardization

13.4 Renaissance Art and Mathematical Perspective

Chapter 14

Early Modern Mathematical Systematization

Chapter 15

The Threshold of Mechanical Computation

Chapter 16

Enlightenment Synthesis and Computational Dreams

Part II

Mathematical Fundamentals

Introduction

The historical journey in Part 1 showed us how humans developed systematic thinking about organized information. Now we need to translate those insights into the precise mathematical language that makes arrays work.

This isn't about learning math for math's sake. Every mathematical concept we explore herefrom basic number properties to linear algebradirectly enables the array operations you'll use in programming. When you understand why multiplication is commutative, you'll understand why certain array optimizations work. When you grasp set theory, you'll see the logic behind array search algorithms. When you work with mathematical functions, you'll understand the elegant relationship between array indices and their values.

We'll build everything from first principles, assuming no advanced mathematical background. But we won't treat mathematics as a collection of arbitrary rules. Instead, we'll see how each concept emerged from the same human drive for systematic organization that we traced in Part 1.

Think of this part as building your mathematical toolkit. Every tool we create here will be used extensively in later parts. By the end, you'll have the mathematical foundation needed to truly understand not just how arrays work, but why they work the way they do.

Chapter 17

The Nature of Numbers and Fundamental Operations

Chapter 18

Real Numbers and Mathematical Completeness

Chapter 19

Fundamental Mathematical Structures

Chapter 20

Functions and Systematic Relationships

Chapter 21

Boolean Algebra and Logical Structures

Chapter 22

Discrete Mathematics and Finite Structures

Chapter 23

Combinatorics and Systematic Counting

Chapter 24

Probability and Systematic Uncertainty

Chapter 25

Linear Algebra and Multidimensional Structures

Chapter 26

Advanced Discrete Structures

Chapter 27

Information Theory and Systematic Representation

Chapter 28

Algorithm Analysis and Systematic Performance

Chapter 29

Mathematical Foundations of Computer Arithmetic

Chapter 30

Advanced Mathematical Structures for Arrays

30.1 Tensor Algebra: Multidimensional Mathematical Objects

30.2 Multilinear Algebra: Systematic Multidimensional Operations

30.3 Fourier Analysis: Systematic Frequency Domain Representation

30.4 Convolution and Systematic Pattern Matching

30.5 Optimization Theory: Systematic Mathematical Improvement

Chapter 31

Mathematical Logic and Formal Systems

Chapter 32

Integration and Mathematical Synthesis

32.1 Connecting Discrete and Continuous Mathematics

32.2 Mathematical Abstraction and Systematic Generalization

32.3 Structural Mathematics: Patterns Across Mathematical Domains

32.4 Mathematical Modeling: Systematic Representation of Real-World Systems

32.5 The Mathematical Mindset: Systematic Thinking for Computational Problems

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

Glossary

Algorithm: A step-by-step procedure...

Array: A data structure consisting...