

Concurrency In C

Programming with POSIX Threads

Software -- Operating Systems.

C++ Concurrency in Action

C++ Concurrency in Action, Second Edition is the definitive guide to writing elegant multithreaded applications in C++. Updated for C++ 17, it carefully addresses every aspect of concurrent development, from starting new threads to designing fully functional multithreaded algorithms and data structures. Concurrency master Anthony Williams presents examples and practical tasks in every chapter, including insights that will delight even the most experienced developer. -- Provided by publisher.

Extreme C

Push the limits of what C - and you - can do, with this high-intensity guide to the most advanced capabilities of C Key FeaturesMake the most of C's low-level control, flexibility, and high performanceA comprehensive guide to C's most powerful and challenging featuresA thought-provoking guide packed with hands-on exercises and examplesBook Description There's a lot more to C than knowing the language syntax. The industry looks for developers with a rigorous, scientific understanding of the principles and practices. Extreme C will teach you to use C's advanced low-level power to write effective, efficient systems. This intensive, practical guide will help you become an expert C programmer. Building on your existing C knowledge, you will master preprocessor directives, macros, conditional compilation, pointers, and much more. You will gain new insight into algorithm design, functions, and structures. You will discover how C helps you squeeze maximum performance out of critical, resource-constrained applications. C still plays a critical role in 21st-century programming, remaining the core language for precision engineering, aviations, space research, and more. This book shows how C works with Unix, how to implement OO principles in C, and fully covers multi-processing. In Extreme C, Amini encourages you to think, question, apply, and experiment for yourself. The book is essential for anybody who wants to take their C to the next level. What you will learnBuild advanced C knowledge on strong foundations, rooted in first principlesUnderstand memory structures and compilation pipeline and how they work, and how to make most out of themApply object-oriented design principles to your procedural C codeWrite low-level code that's close to the hardware and squeezes maximum performance out of a computer systemMaster concurrency, multithreading, multi-processing, and integration with other languagesUnit Testing and debugging, build systems, and inter-process communication for C programmingWho this book is for Extreme C is for C programmers who want to dig deep into the language and its capabilities. It will help you make the most of the low-level control C gives you.

Concurrency in C++

"Concurrency in C++: Writing High-Performance Multithreaded Code" is a comprehensive guide designed to equip programmers with the essential skills needed to develop efficient and robust concurrent applications in C++. The book methodically breaks down the complexities of multithreading, providing a foundation in fundamental concepts such as thread management, synchronization techniques, and memory models. Through detailed explanations and practical examples, readers gain a clear understanding of how to effectively manage multiple threads and ensure data integrity across shared resources. As the book delves into advanced topics, it presents design patterns specifically tailored for concurrency, along with strategies

for optimizing performance in multithreaded applications. It emphasizes real-world examples, illustrating the practical impact of concurrency across various domains, and offers insights into debugging and testing techniques crucial for maintaining reliable software. With an eye on the future, the book also explores new features introduced in C++20 and future trends in concurrent computing, preparing readers to tackle the challenges of modern and emerging computing environments. Written for both novice and experienced developers, this book provides a comprehensive yet accessible approach to mastering concurrency in C++. Whether you're optimizing existing code or creating new multithreaded solutions, "Concurrency in C++" serves as an indispensable resource on the journey to writing high-performance, scalable applications.

Concurrency in C# Cookbook

If you're one of the many developers uncertain about concurrent and multithreaded development, this practical cookbook will change your mind. With more than 75 code-rich recipes, author Stephen Cleary demonstrates parallel processing and asynchronous programming techniques, using libraries and language features in .NET 4.5 and C# 5.0. Concurrency is becoming more common in responsive and scalable application development, but it's been extremely difficult to code. The detailed solutions in this cookbook show you how modern tools raise the level of abstraction, making concurrency much easier than before. Complete with ready-to-use code and discussions about how and why the solution works, you get recipes for using: `async` and `await` for asynchronous operations Parallel programming with the Task Parallel Library The TPL Dataflow library for creating dataflow pipelines Capabilities that Reactive Extensions build on top of LINQ Unit testing with concurrent code Interop scenarios for combining concurrent approaches Immutable, threadsafe, and producer/consumer collections Cancellation support in your concurrent code Asynchronous-friendly Object-Oriented Programming Thread synchronization for accessing data

The Concurrent C Programming Language

“When you begin using multi-threading throughout an application, the importance of clean architecture and design is critical. . . . This places an emphasis on understanding not only the platform’s capabilities but also emerging best practices. Joe does a great job interspersing best practices alongside theory throughout his book.” – From the Foreword by Craig Mundie, Chief Research and Strategy Officer, Microsoft Corporation Author Joe Duffy has risen to the challenge of explaining how to write software that takes full advantage of concurrency and hardware parallelism. In *Concurrent Programming on Windows*, he explains how to design, implement, and maintain large-scale concurrent programs, primarily using C# and C++ for Windows. Duffy aims to give application, system, and library developers the tools and techniques needed to write efficient, safe code for multicore processors. This is important not only for the kinds of problems where concurrency is inherent and easily exploitable—such as server applications, compute-intensive image manipulation, financial analysis, simulations, and AI algorithms—but also for problems that can be speeded up using parallelism but require more effort—such as math libraries, sort routines, report generation, XML manipulation, and stream processing algorithms. *Concurrent Programming on Windows* has four major sections: The first introduces concurrency at a high level, followed by a section that focuses on the fundamental platform features, inner workings, and API details. Next, there is a section that describes common patterns, best practices, algorithms, and data structures that emerge while writing concurrent software. The final section covers many of the common system-wide architectural and process concerns of concurrent programming. This is the only book you’ll need in order to learn the best practices and common patterns for programming with concurrency on Windows and .NET.

Concurrent Programming on Windows

Programming with C++20 teaches programmers with C++ experience the new features of C++20 and how to apply them. It does so by assuming C++11 knowledge. Elements of the standards between C++11 and C++20 will be briefly introduced, if necessary. However, the focus is on teaching the features of C++20. You will start with learning about the so-called big four Concepts, Coroutines, `std::ranges`, and modules. The big four

a followed by smaller yet not less important features. You will learn about `std::format`, the new way to format a string in C++. In chapter 6, you will learn about a new operator, the so-called spaceship operator, which makes you write less code. You then will look at various improvements of the language, ensuring more consistency and reducing surprises. You will learn how lambdas improved in C++20 and what new elements you can now pass as non-type template parameters. Your next stop is the improvements to the STL. Of course, you will not end this book without learning about what happened in the `constexpr`-world.

Programming with C++20

Become a better programmer with performance improvement techniques such as concurrency, lock-free programming, atomic operations, parallelism, and memory management

Key Features

- Learn proven techniques from a heavyweight and recognized expert in C++ and high-performance computing
- Understand the limitations of modern CPUs and their performance impact
- Find out how you can avoid writing inefficient code and get the best optimizations from the compiler
- Learn the tradeoffs and costs of writing high-performance programs

Book Description

The great free lunch of "performance taking care of itself" is over. Until recently, programs got faster by themselves as CPUs were upgraded, but that doesn't happen anymore. The clock frequency of new processors has almost peaked, and while new architectures provide small improvements to existing programs, this only helps slightly. To write efficient software, you now have to know how to program by making good use of the available computing resources, and this book will teach you how to do that. The *Art of Efficient Programming* covers all the major aspects of writing efficient programs, such as using CPU resources and memory efficiently, avoiding unnecessary computations, measuring performance, and how to put concurrency and multithreading to good use. You'll also learn about compiler optimizations and how to use the programming language (C++) more efficiently. Finally, you'll understand how design decisions impact performance. By the end of this book, you'll not only have enough knowledge of processors and compilers to write efficient programs, but you'll also be able to understand which techniques to use and what to measure while improving performance. At its core, this book is about learning how to learn.

What you will learn

- Discover how to use the hardware computing resources in your programs effectively
- Understand the relationship between memory order and memory barriers
- Familiarize yourself with the performance implications of different data structures and organizations
- Assess the performance impact of concurrent memory accessed and how to minimize it
- Discover when to use and when not to use lock-free programming techniques
- Explore different ways to improve the effectiveness of compiler optimizations
- Design APIs for concurrent data structures and high-performance data structures to avoid inefficiencies

Who this book is for

This book is for experienced developers and programmers who work on performance-critical projects and want to learn new techniques to improve the performance of their code. Programmers in algorithmic trading, gaming, bioinformatics, computational genomics, or computational fluid dynamics communities will get the most out of the examples in this book, but the techniques are fairly universal. Although this book uses the C++ language, the concepts demonstrated in the book can be easily transferred or applied to other compiled languages such as C, Java, Rust, Go, and more.

The Art of Writing Efficient Programs

Create robust and scalable applications along with responsive UI using concurrency and the multi-threading infrastructure in .NET and C#

About This Book

Learn to combine your asynchronous operations with Task Parallel Library

Master C#'s asynchronous infrastructure and use asynchronous APIs effectively to achieve optimal responsiveness of the application

An easy-to-follow, example-based guide that helps you to build scalable applications using concurrency in C#

Who This Book Is For

If you are a C# developer who wants to develop modern applications in C# and wants to overcome problems by using asynchronous APIs and standard patterns, then this book is ideal for you. Reasonable development knowledge, an understanding of core elements and applications related to the .Net platform, and also the fundamentals of concurrency is assumed.

What You Will Learn

- Apply general multithreading concepts to your application's design
- Leverage lock-free concurrency and learn about its pros and cons to achieve efficient synchronization between user threads
- Combine your asynchronous operations with Task Parallel Library
- Make your code easier with C#'s

asynchrony support Use common concurrent collections and programming patterns Write scalable and robust server-side asynchronous code Create fast and responsible client applications Avoid common problems and troubleshoot your multi-threaded and asynchronous applications In Detail Starting with the traditional approach to concurrency, you will learn how to write multithreaded concurrent programs and compose ways that won't require locking. You will explore the concepts of parallelism granularity, and fine-grained and coarse-grained parallel tasks by choosing a concurrent program structure and parallelizing the workload optimally. You will also learn how to use task parallel library, cancellations, timeouts, and how to handle errors. You will know how to choose the appropriate data structure for a specific parallel algorithm to achieve scalability and performance. Further, you'll learn about server scalability, asynchronous I/O, and thread pools, and write responsive traditional Windows and Windows Store applications. By the end of the book, you will be able to diagnose and resolve typical problems that could happen in multithreaded applications. Style and approach An easy-to-follow, example-based guide that will walk you through the core principles of concurrency and multithreading using C#.

Mastering C# Concurrency

Learning a language--any language--involves a process wherein you learn to rely less and less on instruction and more increasingly on the aspects of the language you've mastered. Whether you're learning French, Java, or C, at some point you'll set aside the tutorial and attempt to converse on your own. It's not necessary to know every subtle facet of French in order to speak it well, especially if there's a good dictionary available. Likewise, C programmers don't need to memorize every detail of C in order to write good programs. What they need instead is a reliable, comprehensive reference that they can keep nearby. C in a Nutshell is that reference. This long-awaited book is a complete reference to the C programming language and C runtime library. Its purpose is to serve as a convenient, reliable companion in your day-to-day work as a C programmer. C in a Nutshell covers virtually everything you need to program in C, describing all the elements of the language and illustrating their use with numerous examples. The book is divided into three distinct parts. The first part is a fast-paced description, reminiscent of the classic Kernighan & Ritchie text on which many C programmers cut their teeth. It focuses specifically on the C language and preprocessor directives, including extensions introduced to the ANSI standard in 1999. These topics and others are covered: Numeric constants Implicit and explicit type conversions Expressions and operators Functions Fixed-length and variable-length arrays Pointers Dynamic memory management Input and output The second part of the book is a comprehensive reference to the C runtime library; it includes an overview of the contents of the standard headers and a description of each standard library function. Part III provides the necessary knowledge of the C programmer's basic tools: the compiler, the make utility, and the debugger. The tools described here are those in the GNU software collection. C in a Nutshell is the perfect companion to K&R, and destined to be the most reached-for reference on your desk.

C in a Nutshell

The new edition of this classic O'Reilly reference provides clear, detailed explanations of every feature in the C language and runtime library, including multithreading, type-generic macros, and library functions that are new in the 2011 C standard (C11). If you want to understand the effects of an unfamiliar function, and how the standard library requires it to behave, you'll find it here, along with a typical example. Ideal for experienced C and C++ programmers, this book also includes popular tools in the GNU software collection. You'll learn how to build C programs with GNU Make, compile executable programs from C source code, and test and debug your programs with the GNU debugger. In three sections, this authoritative book covers: C language concepts and language elements, with separate chapters on types, statements, pointers, memory management, I/O, and more The C standard library, including an overview of standard headers and a detailed function reference Basic C programming tools in the GNU software collection, with instructions on how use them with the Eclipse IDE

C in a Nutshell

If you're looking to take full advantage of multi-core processors with concurrent programming, this practical book provides the knowledge and hands-on experience you need. The Art of Concurrency is one of the few resources to focus on implementing algorithms in the shared-memory model of multi-core processors, rather than just theoretical models or distributed-memory architectures. The book provides detailed explanations and usable samples to help you transform algorithms from serial to parallel code, along with advice and analysis for avoiding mistakes that programmers typically make when first attempting these computations. Written by an Intel engineer with over two decades of parallel and concurrent programming experience, this book will help you: Understand parallelism and concurrency Explore differences between programming for shared-memory and distributed-memory Learn guidelines for designing multithreaded applications, including testing and tuning Discover how to make best use of different threading libraries, including Windows threads, POSIX threads, OpenMP, and Intel Threading Building Blocks Explore how to implement concurrent algorithms that involve sorting, searching, graphs, and other practical computations The Art of Concurrency shows you how to keep algorithms scalable to take advantage of new processors with even more cores. For developing parallel code algorithms for concurrent programming, this book is a must.

The Art of Concurrency

If you have a working knowledge of Haskell, this hands-on book shows you how to use the language's many APIs and frameworks for writing both parallel and concurrent programs. You'll learn how parallelism exploits multicore processors to speed up computation-heavy programs, and how concurrency enables you to write programs with threads for multiple interactions. Author Simon Marlow walks you through the process with lots of code examples that you can run, experiment with, and extend. Divided into separate sections on Parallel and Concurrent Haskell, this book also includes exercises to help you become familiar with the concepts presented: Express parallelism in Haskell with the Eval monad and Evaluation Strategies Parallelize ordinary Haskell code with the Par monad Build parallel array-based computations, using the Repa library Use the Accelerate library to run computations directly on the GPU Work with basic interfaces for writing concurrent code Build trees of threads for larger and more complex programs Learn how to build high-speed concurrent network servers Write distributed programs that run on multiple machines in a network

Parallel and Concurrent Programming in Haskell

Offers information on how to exploit the parallel architectures in a computer's GPU to improve code performance, scalability, and resilience.

Seven Concurrency Models in Seven Weeks

With threads programming, multiple tasks run concurrently within the same program. They can share a single CPU as processes do or take advantage of multiple CPUs when available. They provide a clean way to divide the tasks of a program while sharing data.

PThreads Programming

Master multithreading and concurrent processing with C++ About This Book Delve into the fundamentals of multithreading and concurrency and find out how to implement them Explore atomic operations to optimize code performance Apply concurrency to both distributed computing and GPGPU processing Who This Book Is For This book is for intermediate C++ developers who wish to extend their knowledge of multithreading and concurrent processing. You should have basic experience with multithreading and be comfortable using C++ development toolchains on the command line. What You Will Learn Deep dive into the details of the how various operating systems currently implement multithreading Choose the best multithreading APIs when designing a new application Explore the use of mutexes, spin-locks, and other synchronization

concepts and see how to safely pass data between threads Understand the level of API support provided by various C++ toolchains Resolve common issues in multithreaded code and recognize common pitfalls using tools such as Memcheck, CacheGrind, DRD, Helgrind, and more Discover the nature of atomic operations and understand how they can be useful in optimizing code Implement a multithreaded application in a distributed computing environment Design a C++-based GPGPU application that employs multithreading In Detail Multithreaded applications execute multiple threads in a single processor environment, allowing developers achieve concurrency. This book will teach you the finer points of multithreading and concurrency concepts and how to apply them efficiently in C++. Divided into three modules, we start with a brief introduction to the fundamentals of multithreading and concurrency concepts. We then take an in-depth look at how these concepts work at the hardware-level as well as how both operating systems and frameworks use these low-level functions. In the next module, you will learn about the native multithreading and concurrency support available in C++ since the 2011 revision, synchronization and communication between threads, debugging concurrent C++ applications, and the best programming practices in C++. In the final module, you will learn about atomic operations before moving on to apply concurrency to distributed and GPGPU-based processing. The comprehensive coverage of essential multithreading concepts means you will be able to efficiently apply multithreading concepts while coding in C++. Style and approach This book is filled with examples that will help you become a master at writing robust concurrent and parallel applications in C++.

Mastering C++ Multithreading

Software -- Programming Languages.

Concurrent Programming in Java

Summary Concurrency in .NET teaches you how to build concurrent and scalable programs in .NET using the functional paradigm. This intermediate-level guide is aimed at developers, architects, and passionate computer programmers who are interested in writing code with improved speed and effectiveness by adopting a declarative and pain-free programming style. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Unlock the incredible performance built into your multi-processor machines. Concurrent applications run faster because they spread work across processor cores, performing several tasks at the same time. Modern tools and techniques on the .NET platform, including parallel LINQ, functional programming, asynchronous programming, and the Task Parallel Library, offer powerful alternatives to traditional thread-based concurrency. About the Book Concurrency in .NET teaches you to write code that delivers the speed you need for performance-sensitive applications. Featuring examples in both C# and F#, this book guides you through concurrent and parallel designs that emphasize functional programming in theory and practice. You'll start with the foundations of concurrency and master essential techniques and design practices to optimize code running on modern multiprocessor systems. What's Inside The most important concurrency abstractions Employing the agent programming model Implementing real-time event-stream processing Executing unbounded asynchronous operations Best concurrent practices and patterns that apply to all platforms About the Reader For readers skilled with C# or F#. About the Book Riccardo Terrell is a seasoned software engineer and Microsoft MVP who is passionate about functional programming. He has over 20 years' experience delivering cost-effective technology solutions in a competitive business environment. Table of Contents PART 1 - Benefits of functional programming applicable to concurrent programs Functional concurrency foundations Functional programming techniques for concurrency Functional data structures and immutability PART 2 - How to approach the different parts of a concurrent program The basics of processing big data: data parallelism, part 1 PLINQ and MapReduce: data parallelism, part 2 Real-time event streams: functional reactive programming Task-based functional parallelism Task asynchronicity for the win Asynchronous functional programming in F# Functional combinators for fluent concurrent programming Applying reactive programming everywhere with agents Parallel workflow and agent programming with TPL Dataflow PART 3 - Modern patterns of concurrent programming applied Recipes and design patterns for successful concurrent programming Building a scalable mobile app with concurrent functional programming

Concurrency in .NET

Software -- Programming Languages.

Expert C Programming

This book is devoted to the most difficult part of concurrent programming, namely synchronization concepts, techniques and principles when the cooperating entities are asynchronous, communicate through a shared memory, and may experience failures. Synchronization is no longer a set of tricks but, due to research results in recent decades, it relies today on sane scientific foundations as explained in this book. In this book the author explains synchronization and the implementation of concurrent objects, presenting in a uniform and comprehensive way the major theoretical and practical results of the past 30 years. Among the key features of the book are a new look at lock-based synchronization (mutual exclusion, semaphores, monitors, path expressions); an introduction to the atomicity consistency criterion and its properties and a specific chapter on transactional memory; an introduction to mutex-freedom and associated progress conditions such as obstruction-freedom and wait-freedom; a presentation of Lamport's hierarchy of safe, regular and atomic registers and associated wait-free constructions; a description of numerous wait-free constructions of concurrent objects (queues, stacks, weak counters, snapshot objects, renaming objects, etc.); a presentation of the computability power of concurrent objects including the notions of universal construction, consensus number and the associated Herlihy's hierarchy; and a survey of failure detector-based constructions of consensus objects. The book is suitable for advanced undergraduate students and graduate students in computer science or computer engineering, graduate students in mathematics interested in the foundations of process synchronization, and practitioners and engineers who need to produce correct concurrent software. The reader should have a basic knowledge of algorithms and operating systems.

Concurrent Programming: Algorithms, Principles, and Foundations

This easy-to-use, fast-moving tutorial introduces you to functional programming with Haskell. You'll learn how to use Haskell in a variety of practical ways, from short scripts to large and demanding applications. Real World Haskell takes you through the basics of functional programming at a brisk pace, and then helps you increase your understanding of Haskell in real-world issues like I/O, performance, dealing with data, concurrency, and more as you move through each chapter.

Real World Haskell

The official book on the Rust programming language, written by the Rust development team at the Mozilla Foundation, fully updated for Rust 2018. The Rust Programming Language is the official book on Rust: an open source systems programming language that helps you write faster, more reliable software. Rust offers control over low-level details (such as memory usage) in combination with high-level ergonomics, eliminating the hassle traditionally associated with low-level languages. The authors of The Rust Programming Language, members of the Rust Core Team, share their knowledge and experience to show you how to take full advantage of Rust's features--from installation to creating robust and scalable programs. You'll begin with basics like creating functions, choosing data types, and binding variables and then move on to more advanced concepts, such as: Ownership and borrowing, lifetimes, and traits Using Rust's memory safety guarantees to build fast, safe programs Testing, error handling, and effective refactoring Generics, smart pointers, multithreading, trait objects, and advanced pattern matching Using Cargo, Rust's built-in package manager, to build, test, and document your code and manage dependencies How best to use Rust's advanced compiler with compiler-led programming techniques You'll find plenty of code examples throughout the book, as well as three chapters dedicated to building complete projects to test your learning: a number guessing game, a Rust implementation of a command line tool, and a multithreaded server. New to this edition: An extended section on Rust macros, an expanded chapter on modules, and appendixes on Rust

development tools and editions.

The Rust Programming Language (Covers Rust 2018)

Master the essentials of concurrent programming, including testing and debugging. This textbook examines languages and libraries for multithreaded programming. Readers learn how to create threads in Java and C++, and develop essential concurrent programming and problem-solving skills. Moreover, the textbook sets itself apart from other comparable works by helping readers to become proficient in key testing and debugging techniques. Among the topics covered, readers are introduced to the relevant aspects of Java, the POSIX Pthreads library, and the Windows Win32 Applications Programming Interface. The authors have developed and fine-tuned this book through the concurrent programming courses they have taught for the past twenty years. The material, which emphasizes practical tools and techniques to solve concurrent programming problems, includes original results from the authors' research. Chapters include: * Introduction to concurrent programming * The critical section problem * Semaphores and locks * Monitors * Message-passing * Message-passing in distributed programs * Testing and debugging concurrent programs. As an aid to both students and instructors, class libraries have been implemented to provide working examples of all the material that is covered. These libraries and the testing techniques they support can be used to assess student-written programs. Each chapter includes exercises that build skills in program writing and help ensure that readers have mastered the chapter's key concepts. The source code for all the listings in the text and for the synchronization libraries is also provided, as well as startup files and test cases for the exercises. This textbook is designed for upper-level undergraduates and graduate students in computer science. With its abundance of practical material and inclusion of working code, coupled with an emphasis on testing and debugging, it is also a highly useful reference for practicing programmers.

Modern Multithreading

Ansible is a simple, but powerful, server and configuration management tool. Learn to use Ansible effectively, whether you manage one server—or thousands.

Is Parallel Programming Hard

The C++11 standard allows programmers to express ideas more clearly, simply, and directly, and to write faster, more efficient code. Bjarne Stroustrup, the designer and original implementer of C++, thoroughly covers the details of this language and its use in his definitive reference, *The C++ Programming Language*, Fourth Edition. In *A Tour of C++*, Stroustrup excerpts the overview chapters from that complete reference, expanding and enhancing them to give an experienced programmer—in just a few hours—a clear idea of what constitutes modern C++. In this concise, self-contained guide, Stroustrup covers most major language features and the major standard-library components—not, of course, in great depth, but to a level that gives programmers a meaningful overview of the language, some key examples, and practical help in getting started. Stroustrup presents the C++ features in the context of the programming styles they support, such as object-oriented and generic programming. His tour is remarkably comprehensive. Coverage begins with the basics, then ranges widely through more advanced topics, including many that are new in C++11, such as move semantics, uniform initialization, lambda expressions, improved containers, random numbers, and concurrency. The tour ends with a discussion of the design and evolution of C++ and the extensions added for C++11. This guide does not aim to teach you how to program (see Stroustrup's *Programming: Principles and Practice Using C++* for that); nor will it be the only resource you'll need for C++ mastery (see Stroustrup's *The C++ Programming Language*, Fourth Edition, for that). If, however, you are a C or C++ programmer wanting greater familiarity with the current C++ language, or a programmer versed in another language wishing to gain an accurate picture of the nature and benefits of modern C++, you can't find a shorter or simpler introduction than this tour provides.

Ansible for DevOps

"The official C++ Core Guidelines provide consistent best practices for writing outstanding modern C++ code, but they aren't organized for easy usage by working developers. In C++ Core Guidelines Explained, expert C++ instructor Rainer Grimm has distilled them to their essence, removing esoterica, sharing new insights and context, and presenting well-tested examples from his own training courses. Grimm helps experienced C++ programmers use the Core Guidelines with any recent version of the language, from C++11 onward. Most of his code examples are written for C++17, with added coverage of newer versions and C++20 wherever appropriate, and references to the official C++ Core Guidelines online"--

A Tour of C++

Summary Functional Programming in C++ teaches developers the practical side of functional programming and the tools that C++ provides to develop software in the functional style. This in-depth guide is full of useful diagrams that help you understand FP concepts and begin to think functionally. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Well-written code is easier to test and reuse, simpler to parallelize, and less error prone. Mastering the functional style of programming can help you tackle the demands of modern apps and will lead to simpler expression of complex program logic, graceful error handling, and elegant concurrency. C++ supports FP with templates, lambdas, and other core language features, along with many parts of the STL. About the Book Functional Programming in C++ helps you unleash the functional side of your brain, as you gain a powerful new perspective on C++ coding. You'll discover dozens of examples, diagrams, and illustrations that break down the functional concepts you can apply in C++, including lazy evaluation, function objects and invocables, algebraic data types, and more. As you read, you'll match FP techniques with practical scenarios where they offer the most benefit. What's inside Writing safer code with no performance penalties Explicitly handling errors through the type system Extending C++ with new control structures Composing tasks with DSLs About the Reader Written for developers with two or more years of experience coding in C++. About the Author Ivan ?uki? is a core developer at KDE and has been coding in C++ since 1998. He teaches modern C++ and functional programming at the Faculty of Mathematics at the University of Belgrade. Table of Contents Introduction to functional programming Getting started with functional programming Function objects Creating new functions from the old ones Purity: Avoiding mutable state Lazy evaluation Ranges Functional data structures Algebraic data types and pattern matching Monads Template metaprogramming Functional design for concurrent systems Testing and debugging

C++ Core Guidelines Explained

"This book is organized around three concepts fundamental to OS construction: virtualization (of CPU and memory), concurrency (locks and condition variables), and persistence (disks, RAIDS, and file systems"-- Back cover.

Functional Programming in C++

When you have questions about C# 7.0 or the .NET CLR and its core Framework assemblies, this bestselling guide has the answers you need. Since its debut in 2000, C# has become a language of unusual flexibility and breadth, but its continual growth means there's always more to learn. Organized around concepts and use cases, this updated edition provides intermediate and advanced programmers with a concise map of C# and .NET knowledge. Dive in and discover why this Nutshell guide is considered the definitive reference on C#. Get up to speed on the C# language, from the basics of syntax and variables to advanced topics such as pointers, operator overloading, and dynamic binding Dig deep into LINQ via three chapters dedicated to the topic Explore concurrency and asynchrony, advanced threading, and parallel programming Work with .NET features, including XML, regular expressions, networking, serialization, reflection, application domains, and security Delve into Roslyn, the modular C# 7.0 compiler-as-a-service

The C++ Standard Library

This book is aimed at readers who are interested in software development but have very little to no prior experience. The book focuses on teaching the core principles around software development. It uses several technologies to this goal (e.g. C, Python, JavaScript, HTML, etc.) but is not a book about the technologies themselves. The reader will learn the basics (or in some cases more) of various technologies along the way, but the focus is on building a foundation for software development. The book is your guided tour through the programming jungle, aiming to provide some clarity and build the foundation for software development skills. The book web site is <https://progbook.org/>

Operating Systems

With a mixture of theory, examples, and well-integrated figures, Embedded Software for the IoT helps the reader understand the details in the technologies behind the devices used in the Internet of Things. It provides an overview of IoT, parameters of designing an embedded system, and good practice concerning code, version control and defect-tracking needed to build and maintain a connected embedded system. After presenting a discussion on the history of the internet and the world wide web the book introduces modern CPUs and operating systems. The author then delves into an in-depth view of core IoT domains including: Wired and wireless networking Digital filters Security in embedded and networked systems Statistical Process Control for Industry 4.0 This book will benefit software developers moving into the embedded realm as well as developers already working with embedded systems.

C# 7.0 in a Nutshell

Your Hands-On Guide to Go, the Revolutionary New Language Designed for Concurrency, Multicore Hardware, and Programmer Convenience Today's most exciting new programming language, Go, is designed from the ground up to help you easily leverage all the power of today's multicore hardware. With this guide, pioneering Go programmer Mark Summerfield shows how to write code that takes full advantage of Go's breakthrough features and idioms. Both a tutorial and a language reference, Programming in Go brings together all the knowledge you need to evaluate Go, think in Go, and write high-performance software with Go. Summerfield presents multiple idiom comparisons showing exactly how Go improves upon older languages, calling special attention to Go's key innovations. Along the way, he explains everything from the absolute basics through Go's lock-free channel-based concurrency and its flexible and unusual duck-typing type-safe approach to object-orientation. Throughout, Summerfield's approach is thoroughly practical. Each chapter offers multiple live code examples designed to encourage experimentation and help you quickly develop mastery. Wherever possible, complete programs and packages are presented to provide realistic use cases, as well as exercises. Coverage includes Quickly getting and installing Go, and building and running Go programs Exploring Go's syntax, features, and extensive standard library Programming Boolean values, expressions, and numeric types Creating, comparing, indexing, slicing, and formatting strings Understanding Go's highly efficient built-in collection types: slices and maps Using Go as a procedural programming language Discovering Go's unusual and flexible approach to object orientation Mastering Go's unique, simple, and natural approach to fine-grained concurrency Reading and writing binary, text, JSON, and XML files Importing and using standard library packages, custom packages, and third-party packages Creating, documenting, unit testing, and benchmarking custom packages

Learn Programming

Threads are a fundamental part of the Java platform. As multicore processors become the norm, using concurrency effectively becomes essential for building high-performance applications. Java SE 5 and 6 are a huge step forward for the development of concurrent applications, with improvements to the Java Virtual Machine to support high-performance, highly scalable concurrent classes and a rich set of new concurrency

building blocks. In Java Concurrency in Practice, the creators of these new facilities explain not only how they work and how to use them, but also the motivation and design patterns behind them. However, developing, testing, and debugging multithreaded programs can still be very difficult; it is all too easy to create concurrent programs that appear to work, but fail when it matters most: in production, under heavy load. Java Concurrency in Practice arms readers with both the theoretical underpinnings and concrete techniques for building reliable, scalable, maintainable concurrent applications. Rather than simply offering an inventory of concurrency APIs and mechanisms, it provides design rules, patterns, and mental models that make it easier to build concurrent programs that are both correct and performant. This book covers: Basic concepts of concurrency and thread safety Techniques for building and composing thread-safe classes Using the concurrency building blocks in java.util.concurrent Performance optimization dos and don'ts Testing concurrent programs Advanced topics such as atomic variables, nonblocking algorithms, and the Java Memory Model

Embedded Software for the IoT

A textbook of C++ examples intended for C programmers. This book is not a starting point for new C++ programmers who do not know C. It is a transition tool for C programmers.

Programming in Go

This title gives students an integrated and rigorous picture of applied computer science, as it comes to play in the construction of a simple yet powerful computer system.

Java Concurrency in Practice

This book is an in-depth introduction to Erlang, a programming language ideal for any situation where concurrency, fault tolerance, and fast response is essential. Erlang is gaining widespread adoption with the advent of multi-core processors and their new scalable approach to concurrency. With this guide you'll learn how to write complex concurrent programs in Erlang, regardless of your programming background or experience. Written by leaders of the international Erlang community -- and based on their training material -- Erlang Programming focuses on the language's syntax and semantics, and explains pattern matching, proper lists, recursion, debugging, networking, and concurrency. This book helps you: Understand the strengths of Erlang and why its designers included specific features Learn the concepts behind concurrency and Erlang's way of handling it Write efficient Erlang programs while keeping code neat and readable Discover how Erlang fills the requirements for distributed systems Add simple graphical user interfaces with little effort Learn Erlang's tracing mechanisms for debugging concurrent and distributed systems Use the built-in Mnesia database and other table storage features Erlang Programming provides exercises at the end of each chapter and simple examples throughout the book.

Concurrency Control and Recovery in Database Systems

C++ for C Programmers

<https://johnsonba.cs.grinnell.edu/~32526040/hcatrvuu/jrojoicoc/wdercayy/rauland+telecenter+v+manual.pdf>
<https://johnsonba.cs.grinnell.edu/~71145421/lmatugg/ashropgr/jcomplitii/its+legal+making+information+technology>
<https://johnsonba.cs.grinnell.edu/~52837376/ygratuhgl/scorroctb/dcomplitim/sharp+lc40le830u+quattron+manual.pdf>
<https://johnsonba.cs.grinnell.edu/~98346122/ygratuhgp/urojoicot/iparlishh/2007+honda+trx450r+owners+manual.pdf>
<https://johnsonba.cs.grinnell.edu/~42853187/wgratuhgk/crojoicor/pspetriy/suzuki+xf650+1996+2001+factory+service>
<https://johnsonba.cs.grinnell.edu/~38975889/vgratuhgj/cshropgm/tinfluinciq/nfhs+football+manual.pdf>
<https://johnsonba.cs.grinnell.edu/~73812848/qlerckl/jroturnm/wspetrid/biological+psychology+11th+edition+kalat>
<https://johnsonba.cs.grinnell.edu/~20575179/kmatugd/fovorflowc/mdercayu/the+murder+of+joe+white+ojibwe+lea>
<https://johnsonba.cs.grinnell.edu/~89154767/grushtp/eovorflowi/kspetrir/american+history+a+survey+11th+edition>
<https://johnsonba.cs.grinnell.edu/~88388340/tmatugi/hroturnu/mspetric/comparative+politics+daniele+caramani.pdf>