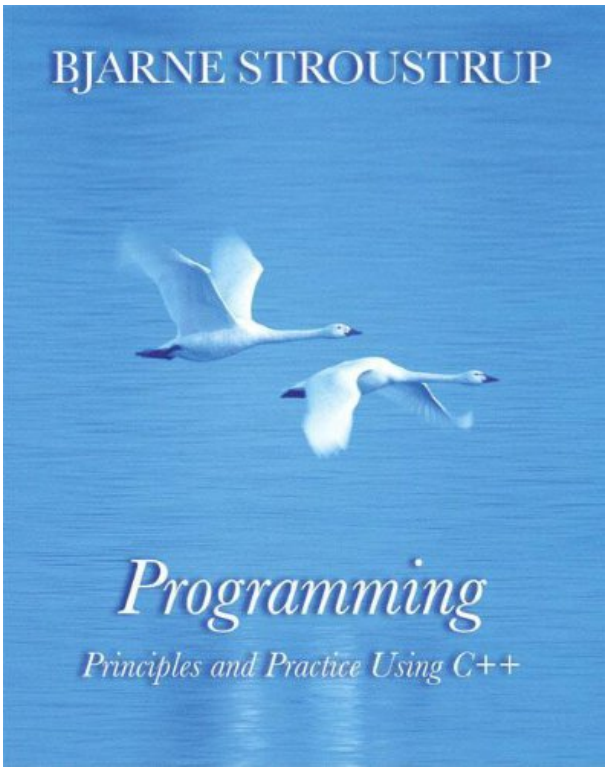
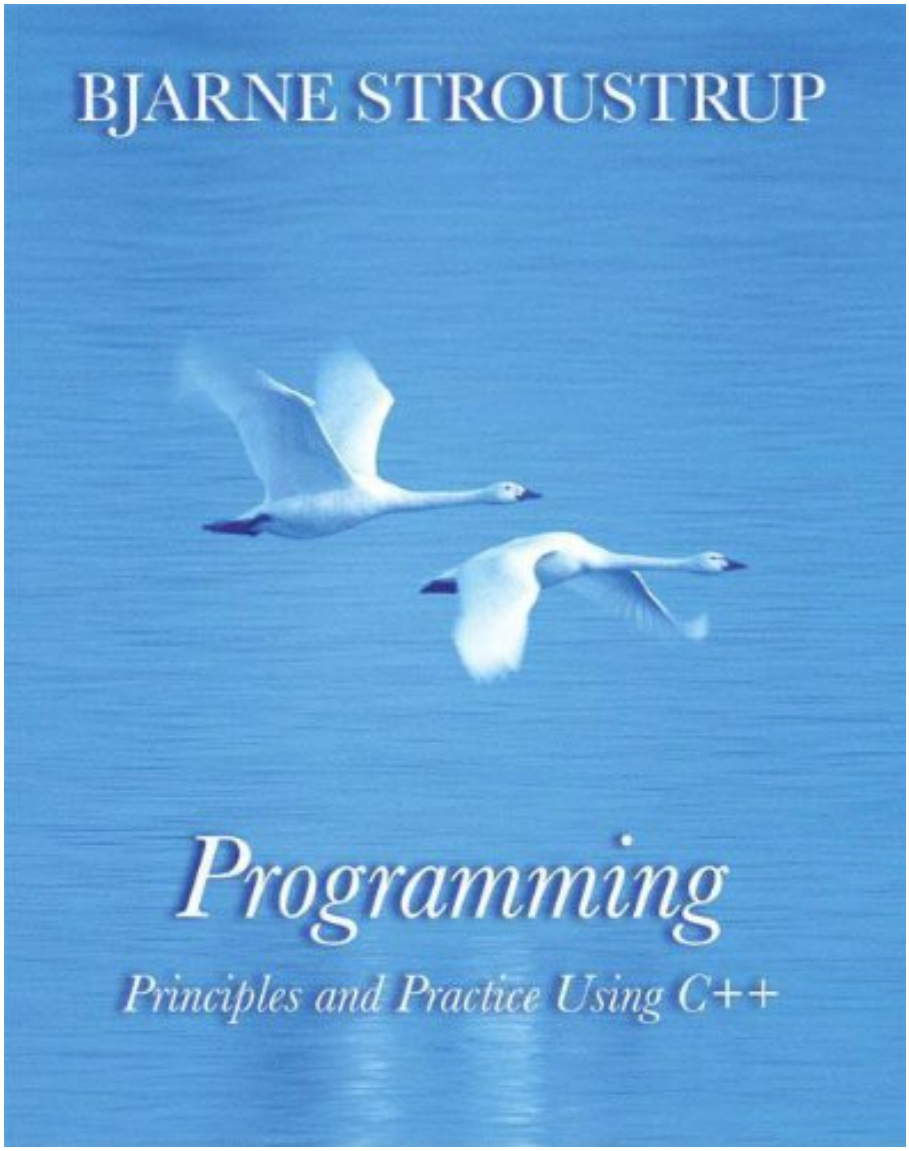
PROGRAMMING: PRINCIPLES AND PRACTICE USING C++ BY BJARNE STROUSTRUP





PROGRAMMING: PRINCIPLES AND PRACTICE USING C++ BY BJARNE STROUSTRUP PDF

Do you think that reading is a crucial task? Find your reasons including is essential. Checking out a book **Programming: Principles And Practice Using C++ By Bjarne Stroustrup** is one component of enjoyable tasks that will make your life top quality better. It is not regarding only what type of e-book Programming: Principles And Practice Using C++ By Bjarne Stroustrup you review, it is not only regarding the amount of publications you check out, it has to do with the behavior. Checking out routine will certainly be a way to make publication Programming: Principles And Practice Using C++ By Bjarne Stroustrup as her or his friend. It will no issue if they invest cash and also spend even more publications to finish reading, so does this e-book Programming: Principles And Practice Using C++ By Bjarne Stroustrup

About the Author

Bjarne Stroustrup is the designer and original implementer of C++, the author of The C++ Programming Language, The Annotated C++ Reference Manual, and The Design and Evolution of C++, and the consulting editor of Addison-Wesley's C++ In-Depth Series. Having previously worked at Bell Labs and AT&T Labs-Research, he currently is the College of Engineering Chair in Computer Science Professor at Texas A&M University. The recipient of numerous honors, including the Dr. Dobb's Excellence in Programming Award (2008), Dr. Stroustrup is a member of the National Academy of Engineering, an AT&T Fellow, an AT&T Bell Laboratories Fellow, an IEEE Fellow, and an ACM Fellow. His research interests include distributed systems, simulation, design, programming techniques, software development tools, and programming languages, and he remains actively involved in the ANSI/ISO standardization of C++. Dr. Stroustrup holds an advanced degree from the University of Aarhus in his native Denmark and a Ph.D. in Computer Science from Cambridge University, England.

“Damn the torpedoes! Full speed ahead.”
—Admiral Farragut

Programming is the art of expressing solutions to problems so that a computer can execute those solutions. Much of the effort in programming is spent finding and refining solutions. Often, a problem is only fully understood through the process of programming a solution for it.

This book is for someone who has never programmed before, but is willing to work hard to learn. It helps you acquire the principles and practical skills of programming using the C++ programming language. My aim is for you to gain sufficient knowledge and experience to perform simple useful programming tasks using the best up-to-date techniques. How long will that take? As part of a first-year university course, you can work through this book in a semester (assuming that you have a workload of four courses of average difficulty). If you work by yourself, don’t expect to spend less time than that (maybe 15 hours a week for 14 weeks).

Three months may seem a long time, but there's a lot to learn and you'll be writing your first simple programs after about an hour. Also, all learning is gradual: each chapter introduces new useful concepts and illustrates them with examples inspired by real-world uses. Your ability to express ideas in code — getting a computer to do what you want it to do — gradually and steadily increases as you go along. I never say "learn a month's worth of theory and then see if you can use it."

Why would you want to program? Our civilization runs on software. Without understanding software you are reduced to believing in "magic" and will be locked out of many of the most interesting, profitable, and socially useful technical fields of work. When I talk about programming, I think of the whole spectrum of computer programs from personal computer applications with GUIs (Graphical User Interfaces), through engineering calculations and embedded system control applications (such as digital cameras, cars, and cell phones), to text manipulation applications as found in many humanities and business applications. Like mathematics, programming — when done well — is a valuable intellectual exercise that sharpens our ability to think. However, thanks to feedback from the computer, programming is more concrete than most forms of math, and therefore accessible to more people. It is a way to reach out and change the world — hopefully for the better. Finally, programming can be great fun.

Why C++? You can't learn to program without a programming language and C++ directly supports the key concepts and techniques used in real-world software. C++ is one of the most widely used programming languages, found in an unsurpassed range of application areas. You find C++ applications everywhere from the bottom of the oceans to the surface of Mars. C++ is precisely and comprehensively defined by a non-proprietary international standard. Quality and/or free implementations are available on every kind of computer. Most of the programming concepts that you will learn using C++ can be used directly in other languages, such as C, C#, Fortran, and Java. Finally, I simply like C++ as a language for writing elegant and efficient code.

This is not the easiest book on beginning programming; it is not meant to be. I just aim for it to be the easiest book from which you can learn the basics of real-world programming. That's quite an ambitious goal because much modern software relies on techniques considered advanced just a few years ago.

My fundamental assumption is that you want to write programs for the use of others, and to do so responsibly providing a decent level of system quality. That is, I assume that you want to achieve a level of professionalism. Consequently, I chose the topics for this book to cover what is needed to get started with real-world programming, not just what is easy to teach and learn. If you need a technique to get basic work done right, I'll describe it, demonstrate concepts and language facilities needed to support the technique, provide exercises for it, and expect you to work on those exercises. If you just want to understand toy programs, you can get along with far less than I present. On the other hand, I won't waste your time with material of marginal practical importance. If an idea is explained here, it's because you'll almost certainly need it.

If your desire is to use the work of others without understanding how things are done and without adding significantly to the code yourself, this book is not for you. If so, please consider if you would be better served by another book and another language. If that is approximately your view of programming, please also consider from where you got that view and whether it in fact is adequate for your needs. People often underestimate the complexity of programming as well as its value. I would hate for you to acquire a dislike for programming because of a mismatch between what you needed and the part of the software reality I describe. There are many parts of the "Information Technology" world that do not require knowledge of programming. This book is aimed to serve those who do want to write nontrivial programs.

Because of its structure and practical aims, this book can also be used as a second book on programming for

someone who already knows a bit of C++ or for someone who programs in another language and wants to learn C++. If you fit into one of those categories, I refrain from guessing how long it will take you to read this book, but I do encourage you to do many of our exercises. This will help you to counteract the common problem of writing programs in older, familiar, styles rather than adopting newer techniques where these are more appropriate. If you have learned C++ in one of the more traditional ways, you'll find something surprising and useful before you reach Chapter 7. Unless your name is Stroustrup, what I discuss here is not "your father's C++."

Programming is learned by writing programs. In this, programming is similar to other endeavors with a practical component. You cannot learn to swim, to play a musical instrument, or to drive a car just from reading a book — you must practice. Nor can you learn to program without reading and writing lots of code. This book focuses on code examples closely tied to explanatory text and diagrams. You need those to understand the ideals, concepts, and principles of programming and to master the language constructs used to express them. That's essential, but by itself, it will not give you the practical skills of programming. For that, you need to do the exercises and get used to the tools for writing, compiling, and running programs. You need to make your own mistakes, and learn to correct them. There is no substitute for writing code. Besides, that's where the fun is!

On the other hand, there is more to programming — much more — than following a few rules and reading the manual. This book is emphatically not focused on "the syntax of C++." Understanding the fundamental ideals, principles, and techniques is essence of a good programmer. Only well-designed code has a chance of becoming part of a correct, reliable, and maintainable system. Also, "the fundamentals" are what lasts: they will still be essential after today's languages and tools have evolved or been replaced.

What about computer science, software engineering, information technology, etc.? Is that all programming? Of course not! Programming is one of the fundamental topics that underlie everything in computer-related fields and has a natural place in a balanced course of computer science. I provide brief introductions to key concepts and techniques of algorithms, data structures, user interfaces, data processing, and software engineering. However, this book is not a substitute for a thorough and balanced study of those topics.

Code can be beautiful as well as useful. This book is written to help you see that, to understand what it means for code to be beautiful and to help you to acquire the principles and practical skills to create such code. Good luck with programming!

A note to students

Of the 1,000++ first-year students we have taught so far using drafts of this book at Texas A&M University, about 60% had programmed before and about 40% had never seen a line of code in their life. Most succeeded, so you can do it too.

You don't have to read this book as part of a course. I assume that the book will be widely used for self study. However, whether you work your way through as part of a course or independently, try to work with others. Programming has an — unfair — reputation as a lonely activity. Most people work better and learn faster when they are part of a group with a common aim. Learning together and discussing problems with friends is not cheating! It is the most efficient — as well as most pleasant — way of making progress. If nothing else, working with friends forces you to articulate your ideas, which is just about the most efficient way of testing your understanding and making sure you remember. You don't actually have to personally discover the answer to every obscure language and programming environment problem. However, please don't cheat yourself by not doing the drills and a fair number of exercises (even if no teacher forces you to do them). Remember: programming is (among other things) a practical skill that you need to practice to

master. If you don’t write code (do several exercises for each chapter), reading this book will become a pointless theoretical exercise.

Most students — especially thoughtful good students — face times where they wonder whether their hard work is worthwhile. When (not if) this happens to you, take a break, re-read the foreword, look at Chapter 1 (“Computers, People, and Programming”) and Chapter 22 (“Ideals and History”). There, I try to articulate what I find exciting about programming and why I consider it a crucial tool for making a positive contribution to the world. If you wonder about my teaching philosophy and general approach, have a look at Chapter 0 (“Notes to the Reader”).

You might find the weight of this book worrying, but it should reassure you that part of the reason for the heft is that I prefer to repeat an explanation or add an example rather than have you search for the one and only explanation. The other major part of the reason is that the last third of the book is “additional material” presented for you to explore only if you are interested in more information about a specific area of programming, such as embedded systems programming, text analysis, or numerical computation.

And please don’t be too impatient. Learning any major new and valuable skill takes time, and is worth it.

A note to teachers

No, this is not a traditional Computer Science 101 course. It is a book about how to construct working software. As such, it leaves out much of what a computer science student is traditionally exposed to (Turing completeness, state machines, discrete math, Chomsky grammars, etc.). Even hardware is ignored on the assumption that students have used computers in various ways since kindergarten. This book does not even try to mention most important CS topics. It is about programming (or more generally about how to develop software) and as such it goes into more detail about fewer topics than many traditional courses. It tries to do just one thing well and Computer Science is not a one-course topic. If this book/course is used as part of a computer science, computer engineering, electrical engineering (many of our first students were EE majors) information science, or whatever program, I expect it to be taught alongside other courses as part of a well-rounded introduction.

Please read Chapter 0 (“Notes to the Reader”) for an explanation of my teaching philosophy, general approach, etc. Please try to convey those ideas to your students along the way.

PROGRAMMING: PRINCIPLES AND PRACTICE USING C++ BY BJARNE STROUSTRUP PDF

Why must pick the problem one if there is easy? Obtain the profit by buying the book **Programming: Principles And Practice Using C++ By Bjarne Stroustrup** here. You will certainly get different means making a bargain and obtain guide Programming: Principles And Practice Using C++ By Bjarne Stroustrup As known, nowadays. Soft documents of guides Programming: Principles And Practice Using C++ By Bjarne Stroustrup end up being very popular among the readers. Are you among them? And right here, we are offering you the brand-new compilation of ours, the Programming: Principles And Practice Using C++ By Bjarne Stroustrup.

In some cases, reading *Programming: Principles And Practice Using C++ By Bjarne Stroustrup* is very dull as well as it will take long time starting from obtaining guide and also begin reading. Nevertheless, in modern-day period, you could take the creating innovation by making use of the internet. By web, you could visit this web page and begin to search for the book Programming: Principles And Practice Using C++ By Bjarne Stroustrup that is needed. Wondering this Programming: Principles And Practice Using C++ By Bjarne Stroustrup is the one that you need, you could opt for downloading. Have you recognized the best ways to get it?

After downloading the soft file of this Programming: Principles And Practice Using C++ By Bjarne Stroustrup, you can start to read it. Yeah, this is so pleasurable while somebody should read by taking their huge books; you remain in your new method by just handle your gizmo. And even you are working in the workplace; you can still utilize the computer to check out Programming: Principles And Practice Using C++ By Bjarne Stroustrup fully. Naturally, it will certainly not obligate you to take lots of web pages. Just page by web page depending on the time that you need to check out Programming: Principles And Practice Using C++ By Bjarne Stroustrup

PROGRAMMING: PRINCIPLES AND PRACTICE USING C++ BY BJARNE STROUSTRUP PDF

An Introduction to Programming by the Inventor of C++

Preparation for Programming in the Real World

The book assumes that you aim eventually to write non-trivial programs, whether for work in software development or in some other technical field.

Focus on Fundamental Concepts and Techniques

The book explains fundamental concepts and techniques in greater depth than traditional introductions. This approach will give you a solid foundation for writing useful, correct, maintainable, and efficient code.

Programming with Today’s C++

The book is an introduction to programming in general, including object-oriented programming and generic programming. It is also a solid introduction to the C++ programming language, one of the most widely used languages for real-world software. The book presents modern C++ programming techniques from the start, introducing the C++ standard library to simplify programming tasks.

For Beginners–And Anyone Who Wants to Learn Something New

The book is primarily designed for people who have never programmed before, and it has been tested with more than 1,000 first-year university students. However, practitioners and advanced students will gain new insight and guidance by seeing how a recognized master approaches the elements of his art.

Provides a Broad View

The first half of the book covers a wide range of essential concepts, design and programming techniques, language features, and libraries. Those will enable you to write programs involving input, output, computation, and simple graphics. The second half explores more specialized topics, such as text processing and testing, and provides abundant reference material. Source code and support supplements are available from the author’s website.

- Sales Rank: #160970 in Books
- Published on: 2008-12-25
- Original language: English
- Number of items: 1
- Dimensions: 9.25" h x 1.54" w x 7.37" l, 4.15 pounds

- Binding: Paperback
- 1272 pages

About the Author

Bjarne Stroustrup is the designer and original implementer of C++, the author of The C++ Programming Language, The Annotated C++ Reference Manual, and The Design and Evolution of C++, and the consulting editor of Addison-Wesley's C++ In-Depth Series. Having previously worked at Bell Labs and AT&T Labs-Research, he currently is the College of Engineering Chair in Computer Science Professor at Texas A&M University. The recipient of numerous honors, including the Dr. Dobb's Excellence in Programming Award (2008), Dr. Stroustrup is a member of the National Academy of Engineering, an AT&T Fellow, an AT&T Bell Laboratories Fellow, an IEEE Fellow, and an ACM Fellow. His research interests include distributed systems, simulation, design, programming techniques, software development tools, and programming languages, and he remains actively involved in the ANSI/ISO standardization of C++. Dr. Stroustrup holds an advanced degree from the University of Aarhus in his native Denmark and a Ph.D. in Computer Science from Cambridge University, England.

“Damn the torpedoes! Full speed ahead.”
—Admiral Farragut

Programming is the art of expressing solutions to problems so that a computer can execute those solutions. Much of the effort in programming is spent finding and refining solutions. Often, a problem is only fully understood through the process of programming a solution for it.

This book is for someone who has never programmed before, but is willing to work hard to learn. It helps you acquire the principles and practical skills of programming using the C++ programming language. My aim is for you to gain sufficient knowledge and experience to perform simple useful programming tasks using the best up-to-date techniques. How long will that take? As part of a first-year university course, you can work through this book in a semester (assuming that you have a workload of four courses of average difficulty). If you work by yourself, don’t expect to spend less time than that (maybe 15 hours a week for 14 weeks).

Three months may seem a long time, but there’s a lot to learn and you’ll be writing your first simple programs after about an hour. Also, all learning is gradual: each chapter introduces new useful concepts and illustrates them with examples inspired by real-world uses. Your ability to express ideas in code — getting a computer to do what you want it to do — gradually and steadily increases as you go along. I never say “learn a month’s worth of theory and then see if you can use it.”

Why would you want to program? Our civilization runs on software. Without understanding software you are reduced to believing in “magic” and will be locked out of many of the most interesting, profitable, and socially useful technical fields of work. When I talk about programming, I think of the whole spectrum of computer programs from personal computer applications with GUIs (Graphical User Interfaces), through engineering calculations and embedded system control applications (such as digital cameras, cars, and cell phones), to text manipulation applications as found in many humanities and business applications. Like mathematics, programming — when done well — is a valuable intellectual exercise that sharpens our ability to think. However, thanks to feedback from the computer, programming is more concrete than most forms of math, and therefore accessible to more people. It is a way to reach out and change the world — hopefully for the better. Finally, programming can be great fun.

Why C++? You can’t learn to program without a programming language and C++ directly supports the key

concepts and techniques used in real-world software. C++ is one of the most widely used programming languages, found in an unsurpassed range of application areas. You find C++ applications everywhere from the bottom of the oceans to the surface of Mars. C++ is precisely and comprehensively defined by a non-proprietary international standard. Quality and/or free implementations are available on every kind of computer. Most of the programming concepts that you will learn using C++ can be used directly in other languages, such as C, C#, Fortran, and Java. Finally, I simply like C++ as a language for writing elegant and efficient code.

This is not the easiest book on beginning programming; it is not meant to be. I just aim for it to be the easiest book from which you can learn the basics of real-world programming. That's quite an ambitious goal because much modern software relies on techniques considered advanced just a few years ago.

My fundamental assumption is that you want to write programs for the use of others, and to do so responsibly providing a decent level of system quality. That is, I assume that you want to achieve a level of professionalism. Consequently, I chose the topics for this book to cover what is needed to get started with real-world programming, not just what is easy to teach and learn. If you need a technique to get basic work done right, I'll describe it, demonstrate concepts and language facilities needed to support the technique, provide exercises for it, and expect you to work on those exercises. If you just want to understand toy programs, you can get along with far less than I present. On the other hand, I won't waste your time with material of marginal practical importance. If an idea is explained here, it's because you'll almost certainly need it.

If your desire is to use the work of others without understanding how things are done and without adding significantly to the code yourself, this book is not for you. If so, please consider if you would be better served by another book and another language. If that is approximately your view of programming, please also consider from where you got that view and whether it in fact is adequate for your needs. People often underestimate the complexity of programming as well as its value. I would hate for you to acquire a dislike for programming because of a mismatch between what you needed and the part of the software reality I describe. There are many parts of the "Information Technology" world that do not require knowledge of programming. This book is aimed to serve those who do want to write nontrivial programs.

Because of its structure and practical aims, this book can also be used as a second book on programming for someone who already knows a bit of C++ or for someone who programs in another language and wants to learn C++. If you fit into one of those categories, I refrain from guessing how long it will take you to read this book, but I do encourage you to do many of our exercises. This will help you to counteract the common problem of writing programs in older, familiar, styles rather than adopting newer techniques where these are more appropriate. If you have learned C++ in one of the more traditional ways, you'll find something surprising and useful before you reach Chapter 7. Unless your name is Stroustrup, what I discuss here is not "your father's C++."

Programming is learned by writing programs. In this, programming is similar to other endeavors with a practical component. You cannot learn to swim, to play a musical instrument, or to drive a car just from reading a book — you must practice. Nor can you learn to program without reading and writing lots of code. This book focuses on code examples closely tied to explanatory text and diagrams. You need those to understand the ideals, concepts, and principles of programming and to master the language constructs used to express them. That's essential, but by itself, it will not give you the practical skills of programming. For that, you need to do the exercises and get used to the tools for writing, compiling, and running programs. You need to make your own mistakes, and learn to correct them. There is no substitute for writing code. Besides, that's where the fun is!

On the other hand, there is more to programming — much more — than following a few rules and reading the manual. This book is emphatically not focused on "the syntax of C++." Understanding the fundamental ideals, principles, and techniques is essence of a good programmer. Only well-designed code has a chance of becoming part of a correct, reliable, and maintainable system. Also, "the fundamentals" are what lasts: they will still be essential after today's languages and tools have evolved or been replaced.

What about computer science, software engineering, information technology, etc.? Is that all programming? Of course not! Programming is one of the fundamental topics that underlie everything in computer-related fields and has a natural place in a balanced course of computer science. I provide brief introductions to key concepts and techniques of algorithms, data structures, user interfaces, data processing, and software engineering. However, this book is not a substitute for a thorough and balanced study of those topics.

Code can be beautiful as well as useful. This book is written to help you see that, to understand what it means for code to be beautiful and to help you to acquire the principles and practical skills to create such code. Good luck with programming!

A note to students

Of the 1,000++ first-year students we have taught so far using drafts of this book at Texas A&M University, about 60% had programmed before and about 40% had never seen a line of code in their life. Most succeeded, so you can do it too.

You don't have to read this book as part of a course. I assume that the book will be widely used for self study. However, whether you work your way through as part of a course or independently, try to work with others. Programming has an — unfair — reputation as a lonely activity. Most people work better and learn faster when they are part of a group with a common aim. Learning together and discussing problems with friends is not cheating! It is the most efficient — as well as most pleasant — way of making progress. If nothing else, working with friends forces you to articulate your ideas, which is just about the most efficient way of testing your understanding and making sure you remember. You don't actually have to personally discover the answer to every obscure language and programming environment problem. However, please don't cheat yourself by not doing the drills and a fair number of exercises (even if no teacher forces you to do them). Remember: programming is (among other things) a practical skill that you need to practice to master. If you don't write code (do several exercises for each chapter), reading this book will become a pointless theoretical exercise.

Most students — especially thoughtful good students — face times where they wonder whether their hard work is worthwhile. When (not if) this happens to you, take a break, re-read the foreword, look at Chapter 1 ("Computers, People, and Programming") and Chapter 22 ("Ideals and History"). There, I try to articulate what I find exciting about programming and why I consider it a crucial tool for making a positive contribution to the world. If you wonder about my teaching philosophy and general approach, have a look at Chapter 0 ("Notes to the Reader").

You might find the weight of this book worrying, but it should reassure you that part of the reason for the heft is that I prefer to repeat an explanation or add an example rather than have you search for the one and only explanation. The other major part of the reason is that the last third of the book is "additional material" presented for you to explore only if you are interested in more information about a specific area of programming, such as embedded systems programming, text analysis, or numerical computation.

And please don't be too impatient. Learning any major new and valuable skill takes time, and is worth it.

A note to teachers

No, this is not a traditional Computer Science 101 course. It is a book about how to construct working software. As such, it leaves out much of what a computer science student is traditionally exposed to (Turing completeness, state machines, discrete math, Chomsky grammars, etc.). Even hardware is ignored on the assumption that students have used computers in various ways since kindergarten. This book does not even try to mention most important CS topics. It is about programming (or more generally about how to develop software) and as such it goes into more detail about fewer topics than many traditional courses. It tries to do just one thing well and Computer Science is not a one-course topic. If this book/course is used as part of a computer science, computer engineering, electrical engineering (many of our first students were EE majors) information science, or whatever program, I expect it to be taught alongside other courses as part of a well-rounded introduction.

Please read Chapter 0 (“Notes to the Reader”) for an explanation of my teaching philosophy, general approach, etc. Please try to convey those ideas to your students along the way.

Most helpful customer reviews

236 of 244 people found the following review helpful.
Not what I was expecting
By Claudio Puviani
For some reasons, I had expected a book on reflections on Stroustrup's philosophy of C++ programming aimed at experienced practitioners. I was quite surprised by the heft of the book, but much more so by the content. It's a book for non-programmers or beginners to teach them how to program with C++ as the vehicle and it's structured for use as a textbook for a first year college course.

Physically, the book is massive, weighing in at over 1200 pages. It is printed on good quality semi-glossy paper and the extensive use of color will remind some of the Deitel & Deitel series, at least superficially.

The prospective student will probably benefit from a comparison of this book to the existing leading tutorial books. The leaders, by popularity or quality, are (in no specific order): Lippman, Lajoie, & Moo's C++ Primer (4th Edition), Eckel's Thinking in C++: Introduction to Standard C++, Volume One (2nd Edition) (Vol 1) and Thinking in C++, Volume 2: Practical Programming, Dietel & Deitel's C++ How to Program (6th Edition), Koenig & Moo's Accelerated C++: Practical Programming by Example (C++ In-Depth Series), Lippman's Essential C++ (C++ In-Depth Series), and Prata's C++ Primer Plus (5th Edition). These all share the common purpose of teaching the C++ language, so an effort is made to cover the features and concepts, with examples that were constructed to illustrate them. This is NOT Stroustrup's approach.

Stroustrup isn't trying to teach the C++ language. He's teaching how to program. C++ is the tool he uses to do so. This isn't a subtle difference. It's the difference between teaching you about a wrench and making up fake car parts to fix with the wrench and teaching you auto repair and giving you a wrench to do so. You still learn the tool as you go along, but it's a side effect.

The overall direction of the book is to teach students how to program solutions to real problems in a way that one would in the real world. Things that other books consider "extraneous to illustrating the principle" aren't swept under the carpet. Inputs are validated. Code is tested. Errors are detected. Exceptions are thrown and caught. They're not incidental details, they're part of the solution, and that's how Stroustrup presents them. Yet, these "details" don't detract from the readability or understandability of the code. In fact, they preempt the stream of "but what about..." questions that students will inevitable have when presented incomplete toy code.

Chapters 6 and 7 are gems. They develop an expression evaluator, walking the student through a tokenizer, parser, and interpreter without bogging the student down with deep theory that will be learned in later courses and is unnecessary to get started (though many will be inspired to go read up on it). Besides showing some interesting and useful techniques, understanding an expression evaluator goes a long way toward understanding programming languages in general.

The discussion on containers and iterators explains how one would go about designing them, not just using them. Once the development of a vector-like container is described, the other standard containers are presented for the student to use. No time is wasted trying to teach data structures, for which other classes and books already exist. The same applies to sorts and other basic algorithms. The standard ones are presented for immediate use by the student.

There are chapters on basic I/O, GUI and graphics (using FLTK), data formatting, and numerical programming (this is my least favorite). There is also some cursory coverage of upcoming C++ features as they are found in boost, such as regular expressions. Because this book deliberately targets beginners, you won't find advanced topics like template metaprogramming. There are entire books (three of them!) dedicated to that.

Finally, there is some brief discussion of the history of C++, on its own and in the context of the evolution of programming languages in general. I would have enjoyed more of this.

If I were to teach a course, this book would be my first choice. A disciplined self-learner would also be well served by this book.

However, it does not try to target those who already know how to program and wish to migrate to C++, though they would doubtless find this book interesting and well written. For them, I would recommend "Accelerated C++" or "Essential C++" to bring them up to speed quickly or ""C++ Primer" to study the language more in depth. For those coming from a language that is conceptually different from C++, the two "Thinking in C++" volumes do a good job of aiding in the paradigm shift. I am deliberately omitting non-tutorial books like the "Effective C++" and "Exceptional C++" series, though they are certainly essential.

Everyone -- beginner and migrating expert -- should avoid C++ How to Program (6th Edition). Notwithstanding the pretty presentation, this book teaches abysmal programming practices, such as blatant and amateurish violations of the Liskov Substitutability Principle.

Naturally, every C++ programmer should own The C++ Programming Language: Special Edition (3rd Edition) and C++ in a Nutshell is a marvelous one-stop reference.

As a side note, there is a long running debate over which language is most suitable for teaching an introduction to programming. C++ is usually one of the first to be eliminated. This book puts C++ back in the running and and shows that it's more about the teacher than it is about the language.

52 of 55 people found the following review helpful.
No better place to start C++ than here...
By Richard Elderton
I had just finished reading Herbert Schildt's book C++: The Complete Reference and had resolved not to read another door stop before devoting much more time to practising the new techniques I had learned. Then I got wind of Bjarne Stroustrup's new book for beginners: Programming Principles and Practice Using C++. Now Dr Stroustrup occupies a very elevated position in the panoply of C++ deities; his words are cast in stone and

he is often referred to as "the creator" of C++ (read: he invented it). Most programming tutorials have shortcomings of one kind or another, so I was intrigued to discover what sort of a job BS had done. I was not disappointed.

Firstly, his approach is not to treat learning C++ as a purely language-technical issue, but to talk about programming as a means to the solving of problems, and use C++ (the most versatile and widely used programming language we have) as a vehicle to do this.

After a dedication to Lawrence Petersen, his collaborator on this project, there is an interesting chapter concerning the place of computer systems in modern life.

Programming is introduced in the conventional way with the simplest concepts, then the learning curve becomes progressively steeper (a feature which is required of a reasonably complete introduction to the subject, even given the 1264 pages of this book).

BS uses several techniques that I had not seen before. All the code is printed in a bold typeface in blue. That makes it easier to distinguish code terms from other, possibly similar words within the body text. He does not use unnecessary spaces in his code. This helps to clarify where spaces are actually required by the syntax as opposed to merely beautifying the code. It also allows more characters per line, but the downside is that the code tends to look more crowded.

Nearly every chapter ends with a set of drills (short exercises), a review of all the new material introduced in the chapter, a list of the new terms, a very comprehensive and well thought out set of more substantial exercises and a postscript giving final thoughts. If students were to take on these exercises in a conscientious way I have no doubt that the learning curve would be flattened to a great extent and they would rapidly gain proficiency in programming.

Having prepared the ground thoroughly, BS raises the level of activity by introducing programming techniques which produce graphical output, and devotes 160 pages in five chapters to it. An independently produced lightweight graphical user interface package called FLTK has to be downloaded and installed for this purpose (its free of charge). FLTK was chosen partly because it is a cross-platform system (cross platform functionality being one of BS's hobby horses, although one which is justified). I found this part of the book a bit tedious, mainly because I am not greatly interested in graphics at present and partly because I did not have the time to play with the system sufficiently.

Two thirds of the way through the book is a refreshing and fascinating chapter dealing with the history of programming and some of the personalities involved; something I had not thought of investigating in any detail before. Colour photographs are another feature of this book which adds to its appeal.

An important theme of the book is the idea that its all too easy to make mistakes when programming, but there are ways to mitigate this. BS owns up and highlights many mistakes he made (some of them deliberate, for pedagogical reasons) when writing programs for the book. I find that both endearing and encouraging. Major sections deal with debugging and system testing, including the recording of run-time.

The last chapter is an introduction to the C programming language. I was very pleased to see that since you cannot go very far in the world of C++ without tripping over branches of C code, and it helps a lot if you can understand it. There are five appendices which provide useful reference material and some extra ideas for anyone who has stayed the course.

The book is supported by some excellent web pages with supporting material including an errata list and well designed tutorial materials for teachers.

I found this book generally very revealing and rate it not only excellent, but inspiring. It provides the means to become a good programmer if you are prepared to do the work, and the encouragement to do so.

32 of 33 people found the following review helpful.
Very good, but not simple.
By Wyatt Willoughby
I agree with Hubert above. This book is great for the "advanced beginner". I too have been through several of the shorter, more to the point, "teach yourself C++"-style books. I am glad that I did those first. I am totally self-taught and unfortunately have no mentor to ask anything. This is the first book that has spent any time on data validation and parsing and has introduced catching errors as part of the bigger picture rather that a discrete subject. This is the way it should be be done.

Chapters 6 introduces parsing. This has been the missing link for me. As soon as you start writing anything beyond the simplest of programs you run into the problem of needing to validate and make sense of user input. You always just knew that there had to be a "standard" way of doing this because the problem is so fundamental, however, the answer is not obvious. I developed a far simpler "home-grown" parser for an ip address calculator program I wrote but using token objects is FAR superior. (which makes me wonder about how many other things I know nothing about) It is not simple and I spent a lot of time reading and trying to truly internalize the chapter and doing the exercises. However, I do wish that more of the answers were published on his site for those of us working at home alone without the benefit of a professor.

Also, graphics and GUIs are given some coverage. This is also unusual for a beginner book. None of the others that I have does this. I think it is important to show that C++ is graphics friendly early on otherwise the ignorant get the impression that only newer languages like VB can handle such things.

So, basically I see this book as a cornerstone in your self-education. It you are working alone it is going to take some time and effort to plow through this monster. (Especially if you do the in-chapter exercises and problems.) But it is a very good book and will likely serve as a nice gateway to more advanced books on the subject.

See all 90 customer reviews...

PROGRAMMING: PRINCIPLES AND PRACTICE USING C++ BY BJARNE STROUSTRUP PDF

After knowing this very easy way to read as well as get this **Programming: Principles And Practice Using C++ By Bjarne Stroustrup**, why don't you inform to others regarding this way? You could tell others to visit this site and also go with browsing them favourite books Programming: Principles And Practice Using C++ By Bjarne Stroustrup As recognized, here are bunches of lists that supply numerous type of books to accumulate. Just prepare couple of time and also internet connections to get the books. You could really delight in the life by checking out Programming: Principles And Practice Using C++ By Bjarne Stroustrup in a really straightforward fashion.

About the Author

Bjarne Stroustrup is the designer and original implementer of C++, the author of The C++ Programming Language, The Annotated C++ Reference Manual, and The Design and Evolution of C++, and the consulting editor of Addison-Wesley's C++ In-Depth Series. Having previously worked at Bell Labs and AT&T Labs-Research, he currently is the College of Engineering Chair in Computer Science Professor at Texas A&M University. The recipient of numerous honors, including the Dr. Dobb's Excellence in Programming Award (2008), Dr. Stroustrup is a member of the National Academy of Engineering, an AT&T Fellow, an AT&T Bell Laboratories Fellow, an IEEE Fellow, and an ACM Fellow. His research interests include distributed systems, simulation, design, programming techniques, software development tools, and programming languages, and he remains actively involved in the ANSI/ISO standardization of C++. Dr. Stroustrup holds an advanced degree from the University of Aarhus in his native Denmark and a Ph.D. in Computer Science from Cambridge University, England.

“Damn the torpedoes! Full speed ahead.”
—Admiral Farragut

Programming is the art of expressing solutions to problems so that a computer can execute those solutions. Much of the effort in programming is spent finding and refining solutions. Often, a problem is only fully understood through the process of programming a solution for it.

This book is for someone who has never programmed before, but is willing to work hard to learn. It helps you acquire the principles and practical skills of programming using the C++ programming language. My aim is for you to gain sufficient knowledge and experience to perform simple useful programming tasks using the best up-to-date techniques. How long will that take? As part of a first-year university course, you can work through this book in a semester (assuming that you have a workload of four courses of average difficulty). If you work by yourself, don’t expect to spend less time than that (maybe 15 hours a week for 14 weeks).

Three months may seem a long time, but there’s a lot to learn and you’ll be writing your first simple programs after about an hour. Also, all learning is gradual: each chapter introduces new useful concepts and illustrates them with examples inspired by real-world uses. Your ability to express ideas in code — getting a computer to do what you want it to do — gradually and steadily increases as you go along. I never say “learn a month’s worth of theory and then see if you can use it.”

Why would you want to program? Our civilization runs on software. Without understanding software you are reduced to believing in "magic" and will be locked out of many of the most interesting, profitable, and socially useful technical fields of work. When I talk about programming, I think of the whole spectrum of computer programs from personal computer applications with GUIs (Graphical User Interfaces), through engineering calculations and embedded system control applications (such as digital cameras, cars, and cell phones), to text manipulation applications as found in many humanities and business applications. Like mathematics, programming — when done well — is a valuable intellectual exercise that sharpens our ability to think. However, thanks to feedback from the computer, programming is more concrete than most forms of math, and therefore accessible to more people. It is a way to reach out and change the world — hopefully for the better. Finally, programming can be great fun.

Why C++? You can't learn to program without a programming language and C++ directly supports the key concepts and techniques used in real-world software. C++ is one of the most widely used programming languages, found in an unsurpassed range of application areas. You find C++ applications everywhere from the bottom of the oceans to the surface of Mars. C++ is precisely and comprehensively defined by a non-proprietary international standard. Quality and/or free implementations are available on every kind of computer. Most of the programming concepts that you will learn using C++ can be used directly in other languages, such as C, C#, Fortran, and Java. Finally, I simply like C++ as a language for writing elegant and efficient code.

This is not the easiest book on beginning programming; it is not meant to be. I just aim for it to be the easiest book from which you can learn the basics of real-world programming. That's quite an ambitious goal because much modern software relies on techniques considered advanced just a few years ago.

My fundamental assumption is that you want to write programs for the use of others, and to do so responsibly providing a decent level of system quality. That is, I assume that you want to achieve a level of professionalism. Consequently, I chose the topics for this book to cover what is needed to get started with real-world programming, not just what is easy to teach and learn. If you need a technique to get basic work done right, I'll describe it, demonstrate concepts and language facilities needed to support the technique, provide exercises for it, and expect you to work on those exercises. If you just want to understand toy programs, you can get along with far less than I present. On the other hand, I won't waste your time with material of marginal practical importance. If an idea is explained here, it's because you'll almost certainly need it.

If your desire is to use the work of others without understanding how things are done and without adding significantly to the code yourself, this book is not for you. If so, please consider if you would be better served by another book and another language. If that is approximately your view of programming, please also consider from where you got that view and whether it in fact is adequate for your needs. People often underestimate the complexity of programming as well as its value. I would hate for you to acquire a dislike for programming because of a mismatch between what you needed and the part of the software reality I describe. There are many parts of the "Information Technology" world that do not require knowledge of programming. This book is aimed to serve those who do want to write nontrivial programs.

Because of its structure and practical aims, this book can also be used as a second book on programming for someone who already knows a bit of C++ or for someone who programs in another language and wants to learn C++. If you fit into one of those categories, I refrain from guessing how long it will take you to read this book, but I do encourage you to do many of our exercises. This will help you to counteract the common problem of writing programs in older, familiar, styles rather than adopting newer techniques where these are more appropriate. If you have learned C++ in one of the more traditional ways, you'll find something surprising and useful before you reach Chapter 7. Unless your name is Stroustrup, what I discuss here is not

“your father’s C++.”

Programming is learned by writing programs. In this, programming is similar to other endeavors with a practical component. You cannot learn to swim, to play a musical instrument, or to drive a car just from reading a book — you must practice. Nor can you learn to program without reading and writing lots of code. This book focuses on code examples closely tied to explanatory text and diagrams. You need those to understand the ideals, concepts, and principles of programming and to master the language constructs used to express them. That’s essential, but by itself, it will not give you the practical skills of programming. For that, you need to do the exercises and get used to the tools for writing, compiling, and running programs. You need to make your own mistakes, and learn to correct them. There is no substitute for writing code. Besides, that’s where the fun is!

On the other hand, there is more to programming — much more — than following a few rules and reading the manual. This book is emphatically not focused on “the syntax of C++.” Understanding the fundamental ideals, principles, and techniques is essence of a good programmer. Only well-designed code has a chance of becoming part of a correct, reliable, and maintainable system. Also, “the fundamentals” are what lasts: they will still be essential after today’s languages and tools have evolved or been replaced.

What about computer science, software engineering, information technology, etc.? Is that all programming? Of course not! Programming is one of the fundamental topics that underlie everything in computer-related fields and has a natural place in a balanced course of computer science. I provide brief introductions to key concepts and techniques of algorithms, data structures, user interfaces, data processing, and software engineering. However, this book is not a substitute for a thorough and balanced study of those topics.

Code can be beautiful as well as useful. This book is written to help you see that, to understand what it means for code to be beautiful and to help you to acquire the principles and practical skills to create such code. Good luck with programming!

A note to students

Of the 1,000++ first-year students we have taught so far using drafts of this book at Texas A&M University, about 60% had programmed before and about 40% had never seen a line of code in their life. Most succeeded, so you can do it too.

You don’t have to read this book as part of a course. I assume that the book will be widely used for self study. However, whether you work your way through as part of a course or independently, try to work with others. Programming has an — unfair — reputation as a lonely activity. Most people work better and learn faster when they are part of a group with a common aim. Learning together and discussing problems with friends is not cheating! It is the most efficient — as well as most pleasant — way of making progress. If nothing else, working with friends forces you to articulate your ideas, which is just about the most efficient way of testing your understanding and making sure you remember. You don’t actually have to personally discover the answer to every obscure language and programming environment problem. However, please don’t cheat yourself by not doing the drills and a fair number of exercises (even if no teacher forces you to do them). Remember: programming is (among other things) a practical skill that you need to practice to master. If you don’t write code (do several exercises for each chapter), reading this book will become a pointless theoretical exercise.

Most students — especially thoughtful good students — face times where they wonder whether their hard work is worthwhile. When (not if) this happens to you, take a break, re-read the foreword, look at Chapter 1 (“Computers, People, and Programming”) and Chapter 22 (“Ideals and History”). There, I try to articulate what I find exciting about programming and why I consider it a crucial tool for making a positive

contribution to the world. If you wonder about my teaching philosophy and general approach, have a look at Chapter 0 (“Notes to the Reader”).

You might find the weight of this book worrying, but it should reassure you that part of the reason for the heft is that I prefer to repeat an explanation or add an example rather than have you search for the one and only explanation. The other major part of the reason is that the last third of the book is “additional material” presented for you to explore only if you are interested in more information about a specific area of programming, such as embedded systems programming, text analysis, or numerical computation.

And please don’t be too impatient. Learning any major new and valuable skill takes time, and is worth it.

A note to teachers

No, this is not a traditional Computer Science 101 course. It is a book about how to construct working software. As such, it leaves out much of what a computer science student is traditionally exposed to (Turing completeness, state machines, discrete math, Chomsky grammars, etc.). Even hardware is ignored on the assumption that students have used computers in various ways since kindergarten. This book does not even try to mention most important CS topics. It is about programming (or more generally about how to develop software) and as such it goes into more detail about fewer topics than many traditional courses. It tries to do just one thing well and Computer Science is not a one-course topic. If this book/course is used as part of a computer science, computer engineering, electrical engineering (many of our first students were EE majors) information science, or whatever program, I expect it to be taught alongside other courses as part of a well-rounded introduction.

Please read Chapter 0 (“Notes to the Reader”) for an explanation of my teaching philosophy, general approach, etc. Please try to convey those ideas to your students along the way.