

Top Twenty Programming Books

by Randy Charles Morin

So it's Boxing Day and you are headed to the bookstore to pick up a few classic programming-books. I know, your wife, siblings and friends didn't buy you any reasonable literature. But then, they wouldn't know what to buy you. Actually, do you know what books you should be buying? Most likely not. Until you've read a book, how do you know you should read it? Well?

Here are my top twenty programming books. These are books that I have read that most influenced me. These books are all classics. Some are specific to a particular technology and some are more general, but each is worth the read.

#20 - Professional DCOM Programming by Richard Grimes

This book covers a very specific topic, that is, DCOM. I myself have written several books on the same subject. I think it can be said that I'm an expert on the technology. This book was one of the original books on the technology and it covered the area extremely accurately and there is very little fluff in this book.

I would have rated this book higher, except that it is horrible hard to read. The writing is very to the point and sometimes has the reader reading paragraphs three and four times in order to comprehend what the author is really trying to say. In later books, Richard Grimes writing style improved dramatically, although I don't think he ever wrote a book as technically brilliant.

I don't think I ever heard of the publisher WROX before this title. I always wondered how much Richard Grimes was responsible for this publisher's success.

<http://www.amazon.com/exec/obidos/ASIN/186100060X/kbcafe>

#19 - The Annotated C++ Reference Manual by Bjarne Stroustrup

Here's another book where a topic expert puts his comments down on paper. Bjarne Stroustrup is of course the father of C++. I wouldn't say that his comments are really all that compelling, but I think it's always smart to read how great men like Stroustrup think.

The book is exactly as the title describes. It's a reference manual of the C++ language with additional comments added from the author of the language.

<http://www.amazon.com/exec/obidos/ASIN/0201514591/kbcafe>

#18 - Fundamentals of Data Structures by Ellis Horowitz and Sartaj Sahni

This is one from my university days. I attended the University of Windsor, Canada. In second year we had a course on data structures. The course followed the writings of an

early edition of this book. More than a decade later, the book is still a great reference manual.

The book has many editions, of which, some are targeted to users of particular languages. The two more popular editions are ones written for the C++ and Pascal languages.

C++ <http://www.amazon.com/exec/obidos/ASIN/0716782928/kbcafe>

Pascal <http://www.amazon.com/exec/obidos/ASIN/0716782634/kbcafe>

#17 - Operating Systems by Andrew Tanenbaum

This is another book from my university days. It was either a second or third year course. The course was awful, but the textbook was a classic. Just so you know how bad the course was, the diskettes distributed by the university and containing the MINIX OS had the stoned virus. If you remember the stoned virus, then you are not in your twenties. It was one of two viruses that I ever got. And I got it twice. Once from the university diskettes for this course and once from a student of the course who had failed to clean all his diskettes. Yes it was you Jake.

Now, the most important thing about this book is that OS. The book contained the entire source code of an OS called MINIX. Another UNIX. Each section of the book addressed a part of the MINIX OS and referred the reader to countless lines of code.

Now here's the question. What became of MINIX? I think you've all heard of LINUX, written by Linus Torvalds. The original versions of LINUX were and could only be compiled from MINIX. Attached, is Linus' announcement more than ten years ago describing his new LINUX OS to the MINIX community. It's a great read. I pulled it off <http://groups.google.com>.

```
From: Linus Benedict Torvalds (torvalds@klaava.Helsinki.FI)
Subject: Free minix-like kernel sources for 386-AT
Newsgroups: comp.os.minix
Date: 1991-10-05 08:53:28 PST
```

```
Do you pine for the nice days of minix-1.1, when men were men and wrote
their own device drivers? Are you without a nice project and just dying
to cut your teeth on a OS you can try to modify for your needs? Are you
finding it frustrating when everything works on minix? No more all-
nighters to get a nifty program working? Then this post might be just
for you :-)
```

```
As I mentioned a month(?) ago, I'm working on a free version of a
minix-lookalike for AT-386 computers. It has finally reached the stage
where it's even usable (though may not be depending on what you want),
and I am willing to put out the sources for wider distribution. It is
just version 0.02 (+1 (very small) patch already), but I've successfully run
bash/gcc/gnu-make/gnu-sed/compress etc under it.
```

```
Sources for this pet project of mine can be found at nic.funet.fi
(128.214.6.100) in the directory /pub/OS/Linux. The directory also
contains some README-file and a couple of binaries to work under linux
(bash, update and gcc, what more can you ask for :-). Full kernel
source is provided, as no minix code has been used. Library sources are only partially
free, so that cannot be distributed currently. The
system is able to compile "as-is" and has been known to work. Heh.
Sources to the binaries (bash and gcc) can be found at the same place in /pub/gnu.
```

```
ALERT! WARNING! NOTE! These sources still need minix-386 to be compiled
(and gcc-1.40, possibly 1.37.1, haven't tested), and you need minix to
set it up if you want to run it, so it is not yet a standalone system
for those of you without minix. I'm working on it. You also need to be
```

something of a hacker to set it up (?), so for those hoping for an alternative to minix-386, please ignore me. It is currently meant for hackers interested in operating systems and 386's with access to minix.

The system needs an AT-compatible harddisk (IDE is fine) and EGA/VGA. If you are still interested, please ftp the README/RELNOTES, and/or mail me for additional info.

I can (well, almost) hear you asking yourselves "why?". Hurd will be out in a year (or two, or next month, who knows), and I've already got minix. This is a program for hackers by a hacker. I've enjoyed doing it, and somebody might enjoy looking at it and even modifying it for their own needs. It is still small enough to understand, use and modify, and I'm looking forward to any comments you might have.

I'm also interested in hearing from anybody who has written any of the utilities/library functions for minix. If your efforts are freely distributable (under copyright or even public domain), I'd like to hear from you, so I can add them to the system. I'm using Earl Chews estdio right now (thanks for a nice and working system Earl), and similar works will be very wellcome. Your (C)'s will of course be left intact. Drop me a line if you are willing to let me use your code.

Linus

PS. to PHIL NELSON! I'm unable to get through to you, and keep getting "forward error - strawberry unknown domain" or something.

Anyhow, you can see how this book is a piece of history. When you read it, keep in mind the book's importance in history. This book was one of the marvels that inspired Linus Torvalds into creating Linux.

Beyond the Operating Systems book, Tanenbaum has written many other books that I also enjoyed reading. After reading this one, you might want to move onto his many other books.

<http://www.amazon.com/exec/obidos/ASIN/0136386776/kbcafe>

#16 - Introduction to Database Systems by C. J. Date

It was back in 1969 that a great computer scientist named Dr. Edgar F Codd began the spiral that was eventually termed the relational model. First with a few internal IBM research papers, than a year later with an article in "*Communications of the ACM*". This last article can be read at the following URL. Read it, or don't read my stuff. Cause you are wasting your time reading my stuff, if you don't read this.

<http://www.acm.org/classics/nov95/>

C. J. Date was to E. F. Codd as Plato was to Socrates. Plato, if you know, wrote plenty about his mentor Socrates. Date also wrote plenty about Codd. Date also wrote plenty with Codd. The first great writing was "Relational Database Writing 1991-94".

<http://www.amazon.com/exec/obidos/ASIN/0201824590/kbcafe>

I haven't read this one yet, but maybe after I do, the book will enter my top twenty list of books. My neglect. Later Date wrote more on the technology than the theoretical. His best writing was Introduction to Database Systems.

<http://www.amazon.com/exec/obidos/ASIN/0201385902/kbcafe>

#15 - Inside OLE by Kraig Brockschmidt

I guess I owe Kraig Brockschmidt a couple dollars. So does every other computer scientist who made money from OLE, COM or DCOM. Brockschmidt's book may have been the catalyst that allowed OLE to thrive. It was a great book. It also led to a large series of books on various related and unrelated subjects. I don't know how much Microsoft may have paid Brockschmidt, but it could not have been enough.

The technology behind this book is ancient and thus the book is no longer in publication. But I still do recommend you comb those used bookstores and find yourself a copy.

<http://www.amazon.com/exec/obidos/ASIN/1556158432/kbcafe>

#14 - Programming Windows by Charles Petzold

Similar to Inside OLE, Microsoft owes a lot to Charles Petzold. His original book Programming Windows 3.1 was likely the catalyst for many programmers including myself writing small W31 utilities. Myself, I wrote an alternative ProgMan that found it's way onto thousands and maybe tens of thousands of desktops. Without Petzold's book, this would not have been possible.

I also wrote used this book in writing a GDI application for a graphics course that netted myself top marks in the course. While everybody else was doing fractals on SGI machines, I wrote a simple VR language using INI files and an interpreter that displayed 3D objects in Windows 3.1. Thanks to Petzold for that A. I could never have done it without him.

Since the Programming Windows 3.1 book Petzold has written follow-ups for Windows 95 and Win32. Of late, he also released a version for C#. The originals are collectors items and the later books still have applications today.

<http://www.amazon.com/exec/obidos/ASIN/157231995X/kbcafe>

#13 - STL Tutorial and Reference Guide by David Musser

David Musser was one of the authors of the STL standard. You can read more about Musser on his homepage.

<http://www.cs.rpi.edu/~musser/>

He later wrote a book on the subject. The book was released prior to the STL standard and thus, it doesn't actually address the entire STL. Specially, it does not discuss the string classes in the STL library. Or at least it didn't in the edition I read.

Again, this is a chance to read into the writings of one of the greatest computer scientist of the last decade.

<http://www.amazon.com/exec/obidos/ASIN/0201379236/kbcafe>

#12 - Inside Visual C++ by David Kruglinski

I had a lot of far ties to David Kruglinski. I use to work for a guy who knew David Kruglinski as a paraglider. I found that funny considering that he was also my favorite author at the time. While still working for this guy, I got this call from Microsoft. They

wanted me to fly down to Seattle with my wife for a week, all expenses paid. I did. That same week, while I was in Seattle, Mr Kruglinski flew into a mountain and died. Weird.

Kruglinski got me my first Visual C++ job. And took me from Borland C++ to Visual C++ in one easy step. I owe him. His books were classics that helped Microsoft's Visual C++ into stardom.

<http://www.amazon.com/exec/obidos/ASIN/1572318570/kbcafe>

#11 - More Effective C++ by Scott Meyers

This was Scott Meyer's second book on C++. This book really made you think about how you were using C++. I don't necessarily agree with everything he wrote, but this book pushed the C++ intellectuals to levels never seen before.

<http://www.amazon.com/exec/obidos/ASIN/020163371X/kbcafe>

#10 - Effective C++ by Scott Meyers

This was Scott Meyer's first book on C++ and pretty much the same as his second book, but slightly better. Obviously, you can pick off the best effective techniques the first time around.

He has recently written another book called Effective STL, which I have yet not read, maybe next year. I'm kind of into books on tape. Laziness, I guess.

<http://www.amazon.com/exec/obidos/ASIN/0201749629/kbcafe>

Both "Effective C++" books are must-reads for all C++ developers, no matter their skill level. The books themselves will serve to increase your skill level. But be wary of the bible factor that such books make possible. In the height of this books popularity, any programming technique could be justified by simply saying that you read it in one of Scott Meyer's book. Since very few had actually taken the time to read the book and most were too lazy to try, coding standards were invented that served the purpose of the author and were justified by simply referring the reader to Scott Meyer's book.

<http://www.amazon.com/exec/obidos/ASIN/0201924889/kbcafe>

#9 - Essential COM by Don Box

Don Box was one of the authors of the SOAP specification. But before he created this great protocol, he wrote a book on COM that was highly technically competent. This book explained COM better than any book explained any subject.

Don Box wrote a second book called Effective COM+. This second book is also a great read, but it is full of technical errata. For example, the author suggests that a remote COM callback could be established in one network cycle and continues on to present an entire section of the book on this premise. Unfortunately, the premise is false and most of the section with it.

<http://www.amazon.com/exec/obidos/ASIN/0201379686/kbcafe>

But in the book "Essential COM", the author presents COM at its most basic level. How it works under the hood. And the book seems extremely accurate. The book was also

extremely easy to read, especially for a technical book. If you do any sort of development on a Microsoft platform, then this book is a must read.

<http://www.amazon.com/exec/obidos/ASIN/0201634465/kbcafe>

#8 - Large Scale C++ Software Design by John Lakos

This book is not as well known as most others on my list, but I hold in very high regard. The book talks about using C++ in large groups, whereas Scott Meyer's books are targeted towards individuals and small groups using the C++ language.

Again, this book is a must read for anybody doing C++ development, but just as Scott Meyer's books suffered from the bible syndrome, so does this book. I had one peer programmer who presented a technique called the "Leveled Architecture" by John Lakos to a group. Unfortunately, he was unaware that someone in the group, that is, me, was familiar the book.

When someone questioned his presentation, the presenter asked him or her if they were familiar with John Lakos' book. Basically, he blew off any question of his presentation with wild references to a technique that nobody was familiar with but himself, and of course me.

After the meeting, I sent him a very good email. I told him how we had a lot in common, that is, both having read and liked John Lakos' book. I also explained that "Leveled Architecture" was a concept in the database systems field and that John Lakos' technique was actually called "Levelization." I worked with him for another six plus months and never heard him refer to John Lakos' book again.

<http://www.amazon.com/exec/obidos/ASIN/0201633620/kbcafe>

#7 - Design Patterns by Erich Gamma

Now here's a classic, I assume you were all waiting for this book. The book presents a set of design patterns, explaining their implementation and when to use them. The book is well written and brought the concept of re-use to a much higher level. Instead of just re-using code, we were now talking about re-using concepts. Each pattern was being written up in every possible language.

The books greatest contribution is how it improved re-use and an understanding of the reusable classes or templates within the development community as a whole. The book is a great read, a must read.

<http://www.amazon.com/exec/obidos/ASIN/0201633612/kbcafe>

#6 - AntiPatterns by Williams Brown

Another great contribution of the "Design Patterns" book by Erich Gamma is that it initiated a series of conceptual books along the same lines. One of these books, I consider to be better than the original.

This book takes the opposite approach as the original book. In this book, the author describes patterns that you should avoid using. It's quite amusing, as you will no doubt have encountered many of these patterns in your day-to-day programming. Some, you

may have encountered many times. I found it very therapeutic laughing at my mistakes and the mistakes of my co-workers.

<http://www.amazon.com/exec/obidos/ASIN/0471197130/kbcafe>

#5 - Object-Oriented Modeling and Design by James Rumbaugh

The story of how UML came about is quite interesting. James Rumbaugh was author of one of the mainstream Object Oriented Modeling techniques called OMT. Grady Booch also presented his own technique. A third technique was called “Use Case” and was authored by Igor Jacobsen.

The development community was torn between the three techniques and a few others. Eventually, Rumbaugh and Booch began appearing together in an attempt to unify the OO techniques. Then Jacobsen was brought into the fold and eventually UML was born.

Unfortunately, UML is not an OO technique. It’s a modeling language, as per its title, Unified Modeling Language. And Rational Rose, which uses UML and all other tools of the genre are tools, they are not OO techniques. But UML and Rational Rose had the affect of destroying all existing OO techniques and making it nearly impossible to introduce new techniques.

Before UML and Rational Rose, I was a follower of the OMT technique. And as such, I still use elements of this technique today. I have also adopted Use Case as a major part of my own personal OO technique. But the root of my OO technique remains with Rumbaugh’s book.

<http://www.amazon.com/exec/obidos/ASIN/0136298419/kbcafe>

#4 - Code Complete by Steve McConnell

Scott Meyer’s books on C++ were much more popular than Code Complete, but Code Complete was clearly a more superior book on coding. I think Scott Meyer’s books were more accepted by the non-Microsoft community.

In Code Complete, the author goes thru great lengths to convince the reader that his technique is proper. He presents a lot of experience and stats to back his positions. Most author’s present their ideas a “do it my way, cause I know better.” This of course leads to a very lengthy book, but well worth the read.

This book is a must read for anybody who writes codes, regardless of the language.

<http://www.amazon.com/exec/obidos/ASIN/1556154844/kbcafe>

#3 - Rapid Development by Steve McConnell

McConnell’s second great book is to development managers as his first great book was to coders. Remember the last five years. Companies were created and destroyed many times in those five years. The reason they did not last, was because they could not complete their target applications before they ran out of money.

If companies are to succeed in the future, then it will not be by hiring a hundred developers in order to develop very complex systems. Managing such large developments is very hard and leads to failures and missed deadlines. Rather the approach will be developing less complex systems, in faster times and with smaller teams of developers. This approach is called RAD, Rapid Application Development. If you are going to adopt the RAD approach, then I suggest you read this book.

<http://www.amazon.com/exec/obidos/ASIN/1556159005/kbcafe>

#2 - Inside the Tornado by Geoffrey Moore

The first seventeen books in this list were technical programming books. The top eighteenth was less about the technique or programming and more about the technique of managing development. Finally, the last two books are about understanding the software development field from the ten thousand foot level.

Geoffrey Moore's books are the best attempt I've found of trying to explain the business cycles in software development. In this book, the author describes technological leaps as tornados. Just as a Tornado strikes quite unpredictably, it is also difficult to time the strike of a technological leap. The author also describes how you can position your business correctly, to become the next eight hundred pound giant after the next tornado.

<http://www.amazon.com/exec/obidos/ASIN/0887308244/kbcafe>

I listened to the book on tapes version of this book and did not actually read it.

#1 - Crossing the Chasm by Geoffrey Moore

And the number one book is from the same author as the number two. This was the author's first successful book. In the book, he describes the growth of an industry having a chasm in the early growth cycle. This chasm causes most software development projects to fail as management panics at the lack of growth. If you can only get to the other side of the chasm, then growth will pick up and the Promised Land will be yours.

What I learnt the most from this author is where to position my career to maximize my successes. It also helps me communicate at the level of the executive. Most of which are familiar with the author's writings.

<http://www.amazon.com/exec/obidos/ASIN/0066620023/kbcafe>