Computer Power User

*Technology That Rocks*

*August 2002• Vol.2 Issue 8*
*Page(s) 108 in print issue*

# Back Door: Q&A With Bjarne Stroustrup

**Jump to first occurrence of:** [BJARNE]


Many times when you see a successful product or standard, a behind-the-scenes device is at least partly responsible. The C++ programming language, which Danish programmer **Bjarne** Stroustrup developed while at AT&T in the early 1980s, is one of those behind-the-scenes devices. C++ is a driving force for many technological and computing advances and products in the past 20 years. Stroustrup, who remains a computer scientist with AT&T Labs and is head of the Large-Scale Programming Research department, recently discussed several aspects of the C++ language with CPU.

**Q: After receiving your Ph.D. in computer science from Cambridge, was designing a programming language like C++ your goal or did you have some other goals?**

**Stroustrup:** My goals had to do with building a distributed operating system. C++ was conceived as a tool for that.

**Q: What sparked your desire to design C++? Are those ideas still relevant as the language evolves?**

**Stroustrup:** My main aim with C++ was to be able to express ideas directly in code and have that code execute with close-to-optimal performance. In other words, to write programs that were both elegant and efficient. That's still my aim, and C++ allows me to do that in many areas that interest me. In particular, I still experiment with distributed systems.

**Q: It seems as though recent changes to C and C++ are driving the two languages further apart. Is this a good thing? Or would you like to see the languages move closer together?**

**Stroustrup:** I would like to see the languages merged. I don't see a philosophical reason not to. I see many practical advantages for the user community from removing incompatibilities, and though a merger would be technically difficult, I

There are people who would contradict every statement in my answer. I'll soon be putting a paper discussing the relationship between C and C++ on my home page [www.research.att.com/~bs/homepage.html].

**Q: Why do you think C++ remains so popular? After developing the language, did you expect to still be talking about C++ 20 years later?**

**Stroustrup:** C++ has amazing expressive power, and many people do know how to use it. Much of today's computing, communications, and commercial infrastructure is C++: for example, Google, Internet Explorer, Photoshop, and critical parts of your phone system. There is a reason for that, and it's not that C++ had the inside track as a proprietary language with superior marketing clout. When developing C++, I was too busy [to think] about anything except how to make it as good as I knew how to and to teach it to anyone who cared to listen or read. If I had thought about it, I guess I would have thought that in 20 years everybody would understand the principles, facilities, and techniques, so that then there would have been no need for me to talk about C++. I would have been very wrong.

**Q: Everyone asks what you would do differently if you could redesign the language. But I'd like to know what features you think were especially good and successful in the original C++ design, features that you'd never change.**

**Stroustrup:** They certainly do (laughing). C++ is focused on classes. A class is meant for expressing a concept. To do that well, a C++ class is a very general and flexible construct. My idea was to make the facilities for defining and using classes so general and flexible that I didn't need to introduce special-purpose facilities. Thus, in C++, a vector is a class, a list is a class, a string is a class, a resource handle can be a class, a thread can be a class, etc. Where other languages have built-in facilities, C++ relies on user-defined classes. The C++ standard library is written in Standard C++. I see class hierarchies and parameterized classes as consequences of wanting general and flexible classes. In the unlikely event that I should design another language, I'd try to strengthen that view. More to the point, I am trying to make that view central to the upcoming revision of the C++ standard.

**Q: What has been the most important change to C++ over the years?**

**Stroustrup:** The addition of templates. The notion of parameterized containers was part of my original conception of a language supporting abstraction. I just didn't know how to design them to be flexible enough and efficient enough until 1988 or so. Improvements in that area continue; generic programming is one of the most active areas of experimentation with C++.

**Q: Do you consider C++ somewhat misunderstood? It seems like people nitpick at perceived deficiencies, yet it's the most widely used language. Is it a case of other developers trying to knock down "the king of the mountain"?**

**Stroustrup:** That's part of the reason, but I don't think that's the main point. For

too many C++ programmers have been taught in a way that leaves out the most effective features and facilities of C++. For example, I recently heard of a C++ class in a university that used a C++ textbook that presents C for the first 500 pages before showing a few examples of classes in the last 100 pages. It has no use of the standard library, there was no use of templates, and no use of exceptions. Basically, the view presented to the poor students was that of a C programmer who had added a few half-digested pieces of C++ at the end as [so-called] advanced material. The likelihood of a student of such a course becoming a capable C++ programmer is very low. If that were C++, I too would be complaining bitterly. Some approaches to C++ focusing on commercial programming environments have similar problems.

In a weird way, that's good news because that allows people to get much better results from C++ without language changes or expensive new tools. I'd like to encourage people who haven't already to have a look at the standard library and try to appreciate the techniques it uses. I'd also like to encourage people to have a look at exception handling. If someone last looked at C++ five or 10 years ago, they'll notice that much has changed, for the better, I think.

For people who think of C++ as "just a better C," I recommend a read of "Learning Standard C++ as a New Language," which can be downloaded from my publications page [www.research.att.com/~bs/papers.html]. There, you can also find other papers about C++. In general, my home page contains a fair amount of information about C++ and links to more C++ sites. For example, you can find the "Standard Library Exception Handling" appendix of "The C++ Programming Language," which should convince even experts that there is more to C++ exception handling than just a few language primitives. That appendix covers programming techniques and standard library guarantees. Using the standard library, and a moderate amount of supposedly advanced features, actually makes C++ easier to learn and to use.

**Q: You've said in the past that you don't like proprietary languages. Could you explain some of the advantages of having a language in the hands of a democratic standards body?**

**Stroustrup:** A proprietary language is typically designed to suit the majority of its owner's customers or a majority of people that the owner would like as customers. A proprietary language tends to pander to such customers. These customers, often being managers and executives, are often not primarily interested in programming, and what the owner corporation wants to sell is typically not primarily a programming language. For example, an owner corporation may take its profits out of hardware sales or operating system sales. A language owner also has a temptation to make incompatible changes to please major customers and, also, to force users to upgrade. Every language owner tries to lock in users by facilities that will not work on alternative suppliers' hardware and/or operating systems. A formal standard offers some minimal protection against that.

A democratic standards body, such as the ISO C++ committee, has other problems, but not those. By definition, an ISO standard body must take everybody's concerns into account. As long as the committee is closely connected

enough for almost everybody rather than something that is ideal for a small group. For me, a very important aspect of C++ as its standards process is that it has not been dominated by concern for the mythical "average user." Over the years, most developers have unusual requirements and needs, so a language deliberately restricted to prevent perceived misuse will get in the way. If you like, you can see that opinion as a result of me working for a company that has greater concern for scale, performance, and reliability than most.

**Q: How much of your current time is spent with C++, making revisions, overseeing its evolution, or simply maintaining it? Are you ever bored with working on C++, or does it remain interesting?**

**Stroustrup:** I spend between a third and half of my time looking at standards issues, answering my C++ related email, giving C++ talks, etc. The rest of my time is spent on more corporate activities and research. Often, those activities have a C++ component also. I do get bored by some aspects of C++, but to compensate, new challenges and opportunities appear all the time. There are also so many interesting people to talk with.

Just now, the C++ standards process is at a critical stage. The ISO standard will soon be five years old, and we need to look toward the future and prepare for a revision. I have a key role in the plans for that revision, called C++0x. I'm the chair of the evolution working group, which will try to define a direction for C++ and will eventually do much of the work on new language features. My personal primary aims are: To make C++ a better language for systems programming and library writing and to make it easier to teach and learn. To do that, and to maintain stability for the huge user community, I propose to be restrained, conservative, and minimalist with language changes while being aggressive and opportunistic with additions to the standard library. My hope is that these aims and policies will lead to a significantly better language.

**Q: What are some other projects you've enjoyed working on in recent years?**

**Stroustrup:** Currently, my personal research has involved a framework for experimenting with search trees and search tree algorithms and a library for the support of C++ program transformation primarily aimed at better support for distributed programming.

Kyle Schurman has been writing about computers and the computing industry for *Smart Computing* and *CPU* magazines for nearly 10 years. (Before you ask, yes, people did use DOS back in those ancient times.)