building block," he said.

The various families of high-temperature superconductors differ in what lies between these layers. In the Y-Ba-Cu-O materials that become superconducting at 93 K, which were the first high-temperature superconductors discovered, it is Cu-O chains that sit between the double pyramidal layers. In the 125 K thallium-based superconductors discovered early this year, it is Tl-O layers sandwiched between the double pyramidal layers. And in the materials fabricated by Bell Labs, the meat of the sandwich is a triple layer—two lead-oxide pyramidal layers on either side of a copper-oxide plane.

The researchers believe the Pb-O/Cu-O/Pb-O layers play the same role in producing superconductivity in the newly discovered superconductors that the Cu-O chains and Tl-O layers play in the earlier materials, Batlogg said. That is, these inner layers accept electrons from the pyramidal Cu-O layers, causing holes (the absence of electrons) in the outer layers. These holes seem necessary to produce superconductivity in the Cu-O layers.

The new family of superconductors has the chemical formula $Pb_2Sr_2ACu_3O_{8+y}$, where A can be either one of the rare earths Y, La, Pr, Nd, Sm, Eu, Gd, Dy, Ho, Tm, Yb, and Lu, or else a mixture of one of these rare earths with Sr or Ca. The Bell Labs scientists say the highest critical temperature they have achieved in this family is 68 K, in $Pb_2Sr_2Y_{0.5}Ca_{0.5}Cu_3O_8$.

Batlogg said the Bell Labs group has seen "clear indications of critical temperatures above 70 K" in some of the new superconductors, but he does not expect the compounds to achieve high enough critical temperatures to be of commercial importance.

In the past 2 years, researchers have found several families of superconductors with critical temperatures above 77 K, the boiling point of nitrogen. These materials are expected to have a number of commercial applications, because they can be cooled with liquid nitrogen instead of the more expensive and difficult to handle liquid helium. The fact that the newly discovered family of superconductors has critical temperatures below 77 K will limit their practical applications.

Preparing the new superconductors is much more complicated than for previously known copper-oxide superconductors, Batlogg said. The Y-Ba-Cu-O materials, for instance, can be prepared simply by grinding together the oxides of yttrium, barium, and copper and baking them, but the best way to make the new materials is to first prepare a copper–oxygen–strontium–rare earth mixture and then react that with PbO at around 900°C. ■ ROBERT POOL

NeXT Embraces a New Way of Programming

The object-oriented approach makes programming easier and faster; NeXT hopes to bring it to the masses

NO DOUBT ABOUT IT: the newly announced NeXT computer is an impressive piece of technology. A product of 3 years' entrepreneurship by Steven Jobs, the dethroned cofounder of Apple Computer, it has explicitly been designed to be an academic's dream machine. It features a built-in network connection, an ultrahigh capacity optical disk for data storage, a blazingly fast central processing unit, the Unix operating system, and more. Best of all, it will sell for only $6500, or about one-half to one-third of what academics are now paying for equally powerful workstations.

For all of that, however, the most intriguing aspect of the NeXT machine is not its hardware, but its software. Even if it stumbles in the marketplace—and most industry analysts agree that the company faces an uphill battle against such entrenched giants as Apple, Sun, and IBM—it still promises to have a lasting influence as the first mass-market computer to embrace "object-oriented" programming: a methodology that is just now emerging from computer laboratories, and that promises to speed software development by factors approaching 10.

Coming at a time when delays, cost overruns, and buggy end products are practically the norm in the software industry, that kind of potential commands attention. "[Object-oriented programming] is one of the few things to come along that could even make a dent in the software bottleneck," says Allen Otis, engineering manager for Servio Logic, Inc., of Beaverton, Oregon, who was chairman of a recent national conference on the subject. "So if it is successful, that is going to make it *the* dominant paradigm for programming."

One way to get a sense of what object-oriented programming is about is to imagine a corporate office. In the standard "procedural" approach to programming, which is embodied in such popular computer languages as Pascal, Fortran, C, and BASIC, one treats the computer like an exceptionally stupid office worker who needs to be told precisely what to do at every step of every task. This does make for a certain efficiency, since the worker can eventually be trained to do the job with no wasted motion whatsoever. But it is also painstaking in the extreme. Make a mistake in describing any one of those steps, which is easy to do, and the machine will obediently start to dun your paid-up customers with bill after bill for $0.00, or some other such idiocy.

In the object-oriented approach, by contrast, the programmer functions more like a high-level executive assembling a team of skilled specialists: "objects" that already know how to handle a variety of tasks. A database object, for example, would be a self-contained piece of code having one preprogrammed method that it can use for sorting a set of records, another method that it can use for extracting the records that meet a given criterion, yet another method for displaying itself in an on-screen window, and so on. The programmer's job is thereby simplified: in principle all he or she has to do is to set up a chain of command among the objects—they operate by passing messages to one another requesting this or that action—and the objects themselves will take care of the rest.

"Instead of people starting on the ground floor with software, they can start on the 10th floor."

The job is simplified even more by the fact that an object-oriented programmer is always reusing and building upon what has gone before, as opposed to designing each new program from scratch. When a new object is needed it is usually defined as a specialization of some existing object, in much the same way that a biologist defines *dog* as a specialization of *mammal*. The new object automatically inherits all the attributes and methods of the old one (*hair*, *warm blood*, *live birth*), so that the programmer only has to add in the unique features (*bark*, *tail-wagging*). Among other things, this makes object-oriented code comparatively easy to debug and to modify, because any given error is usually isolated within a single method of a single object.

The net effect of all this is higher productivity. "As an experiment," says Bruce M. Blumberg, who is in charge of helping outsiders develop software for the NeXT computer, "I wrote a simple text editor" using some of NeXT's predefined objects. "It had multiple windows, scrolling, cut-copy-paste operations, multiple fonts, input-output—and I only had to write about 200 lines of new code." A similar text editor written with the more conventional procedures and subroutines provided on the Macintosh computer requires about 1300 lines of new code, larger by a factor of 6.

Of course, if object-oriented programming is so wonderful, then it is fair to ask why everyone is not using it already. And there are several answers.

The first is that lots of people are using it. Object-oriented programming is certainly not a new idea. Its roots can be traced back into the 1960s, and it had its first real flowering in the mid-1970s with the development of the Smalltalk language at the Xerox Palo Alto Research Center. That language has been commercially available since 1983, and now even has versions that run on the IBM PC and Macintosh. Meanwhile, object-oriented programming in general has become a standard topic for computer science majors. It has likewise become a standard topic in professional seminars and meetings, which have begun to draw increasing numbers of programmers eager to learn more about the subject. And it has already been used on some major new software projects, with a notable example being the symbolic mathematics program *Mathematica* written by University of Illinois physicist Stephen Wolfram, which will be included with the NeXT software package.

Nonetheless, for programmers weaned on the procedural approach—which is to say, the vast majority of programmers—the object-oriented technique has represented a radically different way of thinking about software development. And as long as it required learning a whole new language in addition, the benefits rarely seemed worth the effort. It is perhaps no coincidence that interest in object-oriented techniques has begun to increase at exactly the same time that object-oriented features are being added to well-known procedural languages such as C, Pascal, and Lisp. As one enthusiast points out, object-oriented programming is not a language, but a style.

Just as important, however, the computer community has been slow to embrace the object-oriented technique because its ease of use comes at a price: it is a voracious consumer of a computer's processing power and memory. Fortunately, this appetite has been curbed quite a bit with the advent of more efficient implementations of the technique. Nonetheless, the computational overhead required to manage message-passing, inheritance, and all the rest means that a program written in the object-oriented style will almost never be as fast as the equivalent program written in a procedural style. And that means in turn that the productivity gain promised by object-oriented programming is a practical option only on machines with a lot of horsepower.

Thus the significance of the NeXT computer. Viewed purely as a piece of hardware, it is prototypical of the kind of high-performance, moderately priced workstations that most observers expect to be standard in the 1990s. And yes, it does seem to have ample power for object-oriented programming. Viewed in terms of the software that comes with it, however, it represents the first serious attempt to bring object-oriented programming to a mass market.

In the object-oriented approach, the programmer functions like a high-level executive assembling a team of skilled specialists.

For developers of applications software—desk-top publishing, for example, or statistical analysis packages—the machine will come with an object-oriented version of the C programming language, together with a cluster of 34 predefined objects that the developers can specialize as they wish. At least in this first version of the software, most of these objects are related to on-screen windows, icons, and all the other pieces that go to make up a program's graphical interface. This is no accident. First given wide currency on the graphics-intensive Macintosh computer, which NeXT founder Jobs masterminded in the early 1980s when he was still the head of Apple, these graphical devices have become increasingly popular in recent years as a way of helping users navigate through their programs with minimum training and maximum efficiency. And yet, they have also proved to be very tough to program. On the Macintosh, getting an application's graphical interface right can consume up to 90% of the programmers' time; on the NeXT, that fraction is predicted to drop to 10%. "It should be possible to have much more computational meat in applications programs," says Wolfram.

For the end user, meanwhile, NeXT offers a vivid graphical mechanism for doing object-oriented programming on their own: each object is represented by an icon that can be moved from place to place on the screen with a pointing device known as the "mouse." And the message-passing links are represented as cables that can be connected or disconnected with the click of a button on the mouse. All the programming details are taken care of automatically.

As a demonstration of this capability, the NeXT programmers have created an object whose on-screen display shows a molecule bouncing around inside a piston. Within just a few minutes, a user can link this object to a variety of graphs and gauges, and then use it to illustrate how the motion of the molecule relates to such thermodynamic concepts as temperature and pressure. Ultimately, say NeXT spokesmen, this prototype could be expanded to a much wider palette of physics objects—as well as to similar palettes in other fields—so that any teacher could quickly put together a computer demonstration customized to his or her own purposes. Indeed, NeXT officials say it is quite conceivable that users could become their own programmers for a wide variety of applications. "Instead of people starting on the ground floor" with software, says Jobs, "they can start on the 10th floor."

Although this is by no means the only software environment one can imagine for doing object-oriented programming, it does have the potential for wide-ranging influence. Partly this is because the publicity over the introduction of the NeXT computer should serve to raise the general awareness of the technique. And partly it is because, whether the machine is successful or not, the software is portable to other computers.

The basic NeXT software package, known as NeXTStep, runs on the Unix operating system, which has become a de facto standard for academic computing. In particular, it uses an object-oriented version of Unix known as Mach, which was developed at Carnegie-Mellon University. However, NeXT officials claim that the software can easily be converted to use with other varieties of the operating system—an assertion borne out by the fact that IBM has already licensed NeXTStep for adaptation to its own version of Unix. And they say that the software could in principle be adapted for use with OS/2, the operating system introduced last year for machines in the IBM PC family. Thus, if the software proves as appealing as NeXT hopes, it has a fair bid to become an industry standard.

■ **M. Mitchell Waldrop**