Why

Why Clarion?

Because you demand the best and success is your primary goal!

Whether you are a Programmer, Information Systems Manager, Consulting Client or a Development Executive, Clarion for Windows™ is the most efficient, professional and economical solution in the marketplace today!

You'll find that no other product or company in the industry can make your application development easier. Imagine spending less time on design, programming, implementation, training and maintenance. Simply stated, Clarion will make you and your team more productive and decrease the overall development effort!

At SoftVelocity® their team has provided practical and successful solutions for companies and professionals seeking answers to some important complex questions like:

- How can we decrease the amount of time we spend on software development?
- Can we reduce the time spent on maintaining and enhancing applications?
- Can we access all of the data we want, whenever we want it?
- Will we be able to run faster applications and save precious time?
- Is there an application development tool that is easy to learn and use... Simple, yet powerful? Who is SoftVelocity®, and what is their track record?

Thoughtful questions like these have led our clients to discover the industry's ultimate "secret weapon." So, what are you waiting for? You too will be surprised and pleased with the results we can achieve

The single most important benefit of Clarion is reduced software development effort. Clarion developers complete projects in a third of the time that would be required by Visual Basic (VB), Visual Dot Net, Delphi, or PowerBuilder developers. The reason is simple: the Clarion application generation technology creates source code that programmers don't have to write. Like VB, Delphi and PowerBuilder, Clarion applications reuse code that is already written in the form of custom controls (.VBX and .OCX) or embedded objects (OLE). But unlike VB, Delphi and PowerBuilder, Clarion can generate major portions of an application automatically. This isn't "off the shelf" code. The application generation process can be finely tuned by the developer to create highly complex "made to order" software. This methodology takes a fraction of the time and effort that would be consumed by conventional software development tools.

Clarion Reduces Maintenance

Clarion generated code isn't "pre-written," but it is "pre-tested." A Clarion developer is virtually guaranteed that generated code will compile and run the first time. This is a very different experience from conventional programming which requires a painstaking process of debugging one statement at a time.

Most application generators, such as the Delphi Forms Expert can only be used once. If you change generated source code, the application generator will wipe out your changes if it is used again. That leaves developers with a quandry-- a plain vanilla application or a major maintenance headache.

The Clarion application generator can be used for the entire life cycle of an application no matter how heavily customized it becomes. Clarion applications reside in two repositories: a data dictionary and an application model. The Clarion data dictionary is unique. Most data dictionaries contain information about how data is stored and accessed. The Clarion data dictionary also contains information about how data is displayed and processed. If a database changes or its rules of behavior change or its presentation style changes, a Clarion developer simply corrects the data dictionary, synchronizes the application model, and regenerates the application. What could be easier?

It is just as simple to modify or enhance an application. The same methodology is used to maintain a Clarion application that is used to create it. This is possible because a Clarion developer never changes generated source code. Source code is embedded into the application model so it can be emitted along with the generated code every time the application generator is invoked.

Maintaining applications in design repositories, like the application model and data dictionary, has the side benefit of standardizing and documenting the applications. Clarion applications never become obsolete— they are self-illuminating. If the original developer of an application is not available to make revisions, the project can be assigned to another Clarion developer.

Clarion Accesses Data Anywhere

The data dictionary and application model will instantly communicate the underlying design. Clarion uses proprietary database drivers to make every database look alike. This produces “database neutral” applications that can easily be re-targeted from one database to another. Importantly, there is no cost associated with this portability because Clarion database drivers optimize database operations.

As a result, the performance of a Clarion database application is indistinguishable from an application written in the C language using a native database interface. Only a very talented C or Delphi programmer can match the performance of a Clarion program. It is impossible to produce Clarion performance with Visual Basic or PowerBuilder.

Clarion database access assumes that every database engine contains maximum functionality. Features that are missing in a database engine are supplied by its Clarion database driver. This approach optimizes Clarion database access by choosing efficient access functions and by supplementing or replacing inefficient or missing functions.

The structure of a corporate database is in a constant state of flux. In fact, coping with change is a primary design objective for database engines and database maintenance tools.

Unfortunately, database applications are not as flexible. When a property of a data element changes, all applications using that database must be examined and corrected. This is a tedious process — a process that is completely eliminated with Clarion.

The Clarion Enterprise Edition that we use contains a synchronizing engine that detects, displays, and corrects differences between a Clarion data dictionary and an SQL database. The synchronizer can then modify the data dictionary to match the database or modify the database to match the data dictionary.

The EnterpriseEdition also contains an synchronizing engine that automatically distributes data dictionary changes to database applications. This technology reduces application maintenance to a minimum and ensures that applications always match their underlying database.

Clarion Database Accelerators provide benefits that are indistinguishable from (but less expensive than) the benefits that would be derived by upgrading database server hardware. This technology fine-tunes client applications at run-time, minimizing requests to the database server and overlapping processing at the client with processing at the server. These benefits are automatically incorporated into every Clarion application produced by the application generator Clarion templates are aware of the implicit capacities of their design components. So, a browse template “knows” how many rows can be displayed in a list box and a report template “knows” how many detail lines fit on a page.

Clarion Applications Are Fast

Accordingly, Clarion applications "buffer" data perfectly, minimizing the load on the database server. Local data buffers are then reused as needed making it unnecessary, for example, to access data from the server to redisplay a prior page. Automatic time-outs assure accurate up-to-the-minute information.

Clarion applications can also overlap processing on the client and server by using "asynchronous read-ahead." For example, after receiving a page of data for a browse procedure, the driver immediately requests a second page from the server. By the time the user requests the next page, the data has already arrived. Similarly, a report procedure can be retrieving the next page of data while it formats the current page, eliminating the lesser of the time required to access all of the data or to format all of the output.

Clarion Migrates Legacy Applications

Any developer who has converted a DOS application to Windows will tell you that the application must be completely redesigned. A straight conversion won't "feel" like a Windows program. It won't exhibit "standard Windows behavior." And, of course, it won't use all the fancy "gizmos" that make Windows application easy and fun to use. The developer will probably convince you that the considerable effort spent in cloning a Windows "work-alike" of a DOS program would be wasted.

So, it is not surprising to learn that there are no tools that will automatically convert a DOS program to a Windows program. The intellectual component of the process simply cannot be automated. You have to do it by hand. Unless... you are a Clarion developer. By focusing on the database and application processes, Clarion for Windows™ eliminates much of the effort required to migrate legacy DOS applications to Windows.

It works like this:

- First, you import the database structure into a Clarion data dictionary. Clarion database drivers provide this service for you
- Next you "fuss" over the dictionary, adding file relationships, descriptions, prompt words, column headings, etc...
- Finally, you run the Clarion Application Wizard.

Presto! The wizard creates a complete Windows application that processes your legacy database. In place. Concurrently with your DOS application. It’s true that some custom development will be necessary to finalize the Windows version, but the magnitude of the project has been enormously reduced.

Consider Microsoft Access, a database manager. You program Access by providing a script—a role to play. Then, Access “acts out” your application for you. That’s why Access is so huge. It is full of all the “parts” it must play to perform every application imaginable. Access is also as slow as a snail because it must “interpret” scripts from its language into machine language while your application is running.

Clarion doesn’t work that way. The application generator creates source code written in the Clarion language. The Clarion language has been carefully designed to be understandable to programmers—it can’t be executed by a computer. Turning source code into a form that a computer can understand is called “compiling.” The Clarion optimizing compiler reads Clarion source code and writes extremely efficient machine language. This process occurs during the development cycle.oof,!. Clarion applications run about as fast as a computer can go. For the record, Delphi compiles applications like Clarion. Visual Basic and PowerBuilder are partially compiled. They are slower than Clarion but faster than Access.

Clarion Applications Are Efficient

Clarion applications are also efficient. A typical Clarion database application that prints a report produces an executable file of about 800K. Because the database drivers and print engine are included in the .EXE file, most Clarion applications, along with test data and documentation, can be deployed on a single floppy disk with a single installation step.
^{Vi}owerBuilder

Not so with Delphi, Visual Basic, Visual Dot Net, and PowerBuilder applications. They require multiple installation steps, much more disk storage, and a lot more memory. Delphi uses the Borland Database Engine (BDE) to access databases and the ReportSmith run-time to print reports.

Each requires a separate installation. Similarly, Visual Basic uses the Jet database engine and Crystal Reports run-time. PowerBuilder applications are self-contained but require the services of a huge set of dynamic link library (DLL) files. The resulting footprint of a Delphi, Visual Basic, or PowerBuilder application is well over 8 megabytes.

In other words, Delphi, Visual Basic, and PowerBuilder applications are typically eight to ten times larger than Clarion applications. How can this be?

There are a number of reasons:

BDE and JET are multi-database engines that contain a lot of functionality an application will never use. Clarion drivers work with a single database. If you want to retarget a Clarion application, you simply change to a different driver.

Visual Basic applications require a large run-time interpreter called VBRUN.DLL. Like Access, the VBRUN must play all parts for all applications.

Like VBRUN, the PowerBuilder DLLs must also play all parts for all applications.
ReportSmith was developed by a third party using its own multi-database engine. In other words, the functionality in BDE is duplicated in ReportSmith.

SoftVelocity® writes tighter code and uses more efficient tools than Borland and Microsoft. All three companies use their own compiler technology. And SoftVelocity® compiler technology produces better software.

Of course, Clarion runs on all versions of Windows and produces Windows applications for all versions of Windows. This is possible because the Windows controls, such as tool tips (balloon help) and property sheets (tabbed folders) are built into Clarion and its applications.

Clarion Applications Are Internet Ready

Clarion developers enjoy a 3- to 5-fold productivity advantage while developing Windows database applications. That benefit explodes to a 10- to 20-fold advantage for Internet application development. In a single step, a Clarion application can be converted to a dual Web/Windows applications that can be executed locally or manipulated over the Web by the Internet Explorer browser.

Conventional Internet applications come in two pieces: A Java front-end executed at the Internet browser, and a database requester executed at the Internet server. Developing a Java front-end is an expensive, timeconsuming process because Java is a complex low-level programming language that is beyond the reach of most business programmers. And testing two-piece applications can be a juggling contest, synchronizing debug sessions on a multi-station Intranet.

Not so with Clarion. All Clarion Internet applications use the same ultra-thin, reusable Java front-end supplied with the Clarion Internet Developer's Kit. No development is necessary. The back-end is developed like all Clarion applications—incrementally on a single computer. When complete, the application is converted to a dual Web/Windows executable in a single step. The Web support is transparent when you launch the application from Windows. But when launched by the Clarion Internet Application Broker, the application can be manipulated by any Java-enabled Web browser. Your application looks the same through a browser as it does under Windows.

The ultra-thin reusable Jave client communicates directly with the run-time library embedded in the Clarion application. The Clarion Java Support Library is used with all Clarion Internet applications and need only be downloaded once. Conventional Java front-ends are huge (over 2,000K) application-specific files that must be stored locally on every client or downloaded for each application session.

Clarion Is An Elegant Business Language

Clarion developers love the Clarion language. It is easy to learn, easy to write, and easy to read. Concise but clear, simple yet powerful. The Clarion language was designed specifically for writing Windows business programs. Surprisingly, that makes it unique.

Clarion is the only Windows language with built-in support for database access. It is the only Windows language with built-in support for printing reports. As amazing as this sounds, Clarion is the only Windows language with accurate business math. In fact, Clarion arithmetic produces perfect results of up to 31 decimal digits without introducing rounding errors.

In contrast, Delphi, Visual Basic, and PowerBuilder have no database access commands. None of them support programmable reports, and all three use floating point arithmetic, which is notorious for lost arithmetic significance and precision. No wonder Clarion developers would rather fight than switch.

What good is a powerful development tool, if you don’t know how to use it? Peter Coffee said in a PC Week review that the Clarion documentation set is “exemplary in completeness and organization” and “a model for competing companies.”

Clarion Has World- Class Documentation

In addition to the 2700+ pages of manuals, there is extensive on-line help containing answers to all the most commonly asked questions, along with step-by-step instructions for many common programming tasks, and of course, complete context-sensitive help.
Additional support is provided on the Internet (UseNet Newsgroup -comp.lang.clarion) and on private SoftVelocity® newsgroups where working professional programmers, share their years of experience and expertise with newcomers to Clarion.