Microsoft® Visual C++™
Development System Version 1.0 for Windows™ and Windows NT™

Product Overview

July 1993

For more information contact:

Microsoft Corporation
Eric Lang, (206) 882-8080

Waggener Edstrom
Martin Middlewood, (503) 245-0905

Table of Contents

Introduction

Microsoft® Visual C++™ Development System version 1.0 for Windows™ and Windows NT™, 32-bit Edition is the latest generation of C++ products from Microsoft that significantly shortens the path to writing C++ applications for the Microsoft Windows operating system. Visual C++ 32-bit Edition does this by offering an integrated, task-oriented and 100-percent Windows NT operating system-hosted graphical environment that provides significant productivity benefits and ease of use with complete access to the power of C++.

Design Goals of Visual C++ Standard and Professional Editions

Visual C++ Standard and Professional Editions were introduced in February 1993 for use with the Windows operating system version 3.1.

The major design goals of Microsoft Visual C++ Standard and Professional Editions centered around providing solutions to specific needs that had been identified by the developer community in extensive research. Specifically, Visual C++ was created to be the shortest path to C++ programs for Windows. Ease of use was the number one design goal. Additionally, customers wanted a development tool that protects their investment in learning the tool, as well as their application source code. Visual C++ was designed to accomplish this while providing the smallest and fastest executables in the industry.

Visual C++ Standard and Professional Editions Deliver

Visual C++ Standard and Professional Editions deliver comprehensive solutions to these problems.

- **Shortest path to C++ Programs for Windows.** A completely new integrated graphical tool set, including the Visual Workbench, a combined editor, debugger, graphical browser and a project management system designed to be intuitive allowed programmers to easily control project options for the compiler and linker. All options can easily be set through dialog boxes. Extensive online help facilities were also provided.

  Two wizards, the App Wizard and Class Wizard, combine to assist the developer in setting up new projects and in managing the derivation of classes from the MFC 2.0 application framework. Developers can work with the application framework to create new classes and member functions, allowing Class Wizard to manage the mundane details of hooking the new classes up to user interface components.

  App Studio is a completely integrated resource editor that combines new visual editors for all of the standard Windows resources into one easy-to-use tool featuring drag-and-drop and point-and-click simplicity. The streamlined integration with Visual Workbench and Class Wizard make it the easiest way to connect user interface elements to C++ classes and application code.

- **Safest Investment.** Microsoft Foundation Class (MFC) 2.0 improves upon the popular C++ development set for Windows with the Microsoft Foundation Class Library 1.0. Applications written using MFC 1.0 move easily to MFC 2.0 to take advantage of the new high level features. The new library provides architecture classes and classes for advanced features such as Toolbars, Status Bars, Printing and Print Preview, Splitter Windows and more.

  In addition, MFC was designed to coexist with existing C and C++ code so that developers can add new features to existing applications without having to completely rewrite them.

- **Smallest and Fastest Code.** Building on the legacy of code generation quality set with previous versions of Microsoft compilers, the optimization technology in Visual C++ Professional Edition is second to none, providing highly tuned applications with unlimited flexibility. The developer controls exactly which optimizations are used. Visual C++ Professional Edition also added new switches to automatically create either the smallest, or the fastest executable, avoiding the complications of custom optimizations where such fine control is not necessary.

  The entire Visual C++ family is the result of listening to the concerns of developer customers. Making C++ and programming for Windows easy is the number one concern. All versions of Visual C++ have been designed with these goals in mind.

What's New For The 32-bit Edition

Visual C++ 32-bit Edition version 1.0, formally announced in August 1993, includes SDK tools and advanced optimizations for the Win32™ and Win32s™ application program interface (API) and is designed to be hosted under the Windows NT operating system. Like the other Visual C++ editions, the 32-bit Edition was designed with specific customer-driven design goals. The specific design goals for the 32-bit Edition include the following:

- **Shortest Path to 32-bit Windows.** Bringing the ease-of-use features from the 16-bit product to 32-bits was crucial. Streamlined integration and wizards can be used to assist the developer in creating powerful 32-bit applications for both Windows NT and Windows 3.1. The power of the Windows NT host operating system can be exploited to allow seamless background compilation and simultaneous multiple project builds. In addition, the developer is assisted in all phases of 32-bit development, from project creation to debugging.

- **Safest Investment.** Visual C++ 32-bit Edition is the safest investment in C++ development tool technology because it provides an easy 16-bit to 32-bit migration path from previous Microsoft C++ products. It enables developers to use existing NMAKE files, and to work with existing C and C++-based projects. Visual C++ 32-bit provides a way for developers to write applications from a single source code base that will run on both Windows 3.1 and the new Windows NT operating system, taking advantage of the performance benefits of 32-bit programming on both platforms.

Visual C++ 32-bit Edition Delivers

Visual C++ 32-bit Edition is the latest addition to the Visual C++ family. It delivers on each of these objectives. Visual C++ 32-bit Edition uniquely addresses these objectives by providing the ease-of-use features introduced in the Standard and Professional Editions, and incorporating enhancements for support of 32-bit programming and the Windows NT host platform.

Shortest Path to 32-bit Windows

- **IDE Enhancements.** The Visual Workbench and integrated debugger have been enhanced with new features for 32-bit programming while providing the same intuitive interface as the 16-bit products. Developers already familiar with Visual C++ Standard or Professional Editions will be right at home working with the new 32-bit edition. However, to improve on the ease of use, we added several new enhancements, including the following:
  - IDE is a completely 32-bit application itself and takes advantage of the advanced features of the Windows NT operating system, including preemptive multitasking and multithreading. This allows the developer to continue working in project files while builds are progressing in the background. Additionally, it is possible to have multiple builds occurring simultaneously.
  - Integrated Profiler makes profiling just a few mouse clicks away. It has tremendous improvement in its ease of use and enables developers to see potential performance problem areas in their code at a glance.
  - Find-in-files utility allows global searching through source files for any text string. The integration of this feature allows the developer to go directly to the place in the source code the references are found by simply clicking the result in the output window. This is just one of many ways that the Visual Workbench allows virtually unlimited navigation of complex project files, dramatically improving programmer productivity when navigating through the Win32 API.
  - Enhanced support for third-party tool integration — tools can now direct their output to the IDE output window. Tools can easily be added to the Tools menu for easy access. Again, the improved integration provides a more seamless environment for the developer, making it much easier to take advantage of capabilities offered by third-party tools.

- **Debugger Enhancements.** The IDE debugger in Visual C++, 32-Bit Edition, has been enhanced with the features developers have asked for most. In addition to all of the features included in the Standard and Professional Editions, the 32-bit Edition includes the following:
  - New memory window — displays current memory contents.
  - Support for Structured Exception Handling — allows the developer to control the action of the debugger when an exception is thrown. Options include stop if unhandled, and stop always. User defined exceptions can be created. This configurability allows the developer greater control of the application during the debugging stage, and assists in locating coding errors.
  - Support for threads — allows the developer to view the current execution state of all threads in the target application, pause or resume thread execution, view addresses or functions and begin additional debugging tasks such as obtaining a stack trace. Double clicking opens the appropriate source file and positions the highlight on the line of execution. This can dramatically ease the task of debugging a multithreaded application because it is possible to easily follow the

execution path of any given thread.

- **New Spy++ Windows Analysis Tool.** Spy++ is the greatly enhanced window message spy utility for Visual C++ 32-bit Edition that allows a developer to quickly discover a great deal of information about his or her Windows-based application. The display of window messages now shows decoded parameters as well as return values. Flexible "filters" allow precise control over the messages, windows and other information displayed. In addition to the view that shows the messages going to a window, three different "Tree" view types show the hierarchical relationships of windows, processes and threads in the system. Four other "Detail" view types show details about specific windows, processes, threads and window classes in the system. Any number of these different views can be displayed at once. Links are used throughout the Spy++ views to allow a developer to quickly get more information about the items that are shown.

- **Compiler Enhancements.** The Visual C++ 32-bit Edition compiler is a proven technology. More than 100 million lines of code were run through this compiler during the development process. In fact, it is the compiler that was used to build Windows NT itself. New code generation for i486™ and Pentium™ processors, coupled with advanced optimization for the new Pentium superscalar architecture, assures the fastest and smallest code possible for the highest performing application. Microsoft worked closely with Intel in implementing this advanced optimization technology. The new i486 and Pentium optimizations offer dramatic performance improvements for target applications.

  The 32-bit Edition also continues support for precompiled headers, and all compiler options can be easily controlled from the Visual Workbench.

Safest Investment — Still

- **MFC 2.0.** The Microsoft Foundation Class Library 1.0 became a standard for C++ application development for Windows. Microsoft committed to ensuring that applications written to the MFC standard would move forward to our newer operating systems. MFC 1.0 has been shipping with the Win32 prerelease SDK, and many 32-bit applications have been written using it. MFC 2.0 builds on the foundation established with MFC 1.0 by adding architecture classes and classes for such advanced user interface features as Splitter Windows, Toolbars, Status Bars, Printing and Print Preview and more. Applications written using MFC 1.0 can easily be moved to MFC 2.0, preserving the developer's investment.

  Now, 16-bit applications written using either MFC 1.0 or MFC 2.0 can easily be moved to 32-bits, providing improved application performance. The resulting 32-bit application can run on Windows 3.1 or Windows NT, allowing developers to maintain a single source code base for both operating systems.

- **Windows 3.1 Targetting.** The Visual C++ 32-bit Edition includes the Win32 SDK components necessary for building applications for Windows NT. It also includes Win32s, a product that allows 32-bit Win32-based applications to run binary-compatibly under both Windows NT and Windows operating system version 3.1. By

writing applications to the Win32 API using Win32s, developers can create single source base 32-bit applications that will run on both Windows 3.1 and Windows NT. Because of the flat address model of 32-bit programming, it is often easier to write 32-bit applications, and these applications often will run faster than comparable 16-bit applications, even when running on Windows 3.1.

Win32s is a complete development solution, including a debugger and command line tools that operate on the Windows 3.1 target. A developer can edit, build and debug the application under Windows NT, and then move the source code to a Windows 3.1 target for final performance tuning and debugging.

Applications written using MFC 2.0 are fully compatible with Win32s.

- **Exploits CD-ROM with Books Online.** Visual C++ 32-bit Edition is the first PC based development tool to fully exploit the power of CD-ROM media. In addition to the software and online help, more than 10,000 pages of documentation has been cross referenced and indexed into a new Books Online system. Developers have instant access to the users' guides, tutorials, language references and API references. The new viewer technology shipped with Visual C++ 32-bit Edition includes extensive browsing capabilities and an enhanced full-text search feature so that even topics that cannot be found through the extensive indexing can easily be found using wild cards, pattern matching, etc. Books Online includes all of the graphics and source code samples in the available printed documentation and is truly information at your fingertips.

For those developers who also wish printed documentation, it can be obtained at additional cost.

Figure 1

Visual C++ 32-bit edition, also allows developers to run the development environment directly from the CD-ROM, saving up to 75 megabytes of hard disk space for developers who use this option.

As summarized in the chart below, Visual C++, 32-Bit Edition is the product to choose if you wish to create powerful 32-bit applications for Windows 3.1 and Windows NT.

	Visual C++ Standard Edition	**Visual C++ Professional Edition**	**Visual C++ 32-bit Edition**
Hosted on	Windows 3.1	Windows 3.1	Windows NT
Targets	Windows 3.1 (Win 16)	Windows 3.1 (Win 16) MS-DOS® (DOS 16)	Windows 3.1 (Win32s) Windows NT (Win32)

Moving to 32-bit C++ Programming in Windows

As mentioned above, the difficulty of moving to C++ programming in the Windows environment is a major issue for developers. The fastest way to show how the Visual C++ development system helps programmers overcome these difficulties is to briefly describe the process needed to create a new application that uses the new features of Microsoft Foundation Class version 2.0.

In the next few pages, a series of steps demonstrate how Visual C++ 32-Bit Edition can speed the development of 32-bit C++ applications for the Windows or Windows NT environments. An overview of these steps are as follows:

1. Visual Workbench: Starts the integrated development environment.
2. AppWizard: Creates a new "skeleton" application that supports Microsoft Foundation Class 2.0.
3. App Studio: Creates, edits and browses application resources.
4. ClassWizard: Connects user interface elements to C++ code.
5. Microsoft Foundation Class 2.0: Leverages the architecture and high-level abstractions of the application framework.
6. Visual Workbench: Builds and debugs the application.

The following "demo" covers only a few of the more important features. For a more comprehensive discussion of the features of Visual C++, see Appendix A. For an actual programming exercise, see Appendix B.

Step 1: Start Visual Workbench

Double-clicking the Visual Workbench icon launches the Visual C++ completely new, 100-percent Windows-hosted development environment. Visual Workbench is a fully integrated environment that enables programmers to edit, build, debug and browse program code in a visually oriented manner. It includes a source code editor, compiler/linker, debugger and class browser.

Visual Workbench also provides completely seamless access to all other Visual C++ tools, including App Studio for resource editing and ClassWizard for connecting user interface elements to application code.

Visual Workbench allows programmers to build or edit code in the background. For example, you can click the build button to compile and link your application and then use the editor to write a new function while the build is completed as a background task.

µ §

Figure 2. Visual Workbench

Step 2: AppWizard

To create a new application, you select AppWizard from the Visual Workbench Project menu. The purpose of AppWizard is to create a full set of "skeleton" application source code files and project files that give programmers a "jump start" into programming with Visual C++ using Microsoft Foundation Class 2.0. By clicking check boxes in an AppWizard dialog (see figure three), you can choose which Microsoft Foundation Class 2.0 features you want the application

to contain, including:

- Print and Print Preview
- Multiple document interface (MDI) support
- OLE client support
- Context-sensitive Help
- Toolbar capabilities

The key productivity benefit of AppWizard is that it reuses a substantial amount of prefabricated code in the application framework. All the initial source code for the selected features is included in the application's project files. It takes only a few seconds to create the skeleton application, which you can build and run immediately. Because so much functionality is included, creating the skeleton application manually would have taken considerably longer.
Additionally, the application automatically supports the new Microsoft Foundation Class 2.0 document and view architecture (described in Appendix A and the Microsoft Foundation Class 2.0 white paper).

AppWizard is therefore the fastest way to get started with Microsoft Foundation Class 2.0 programming because it creates baseline Microsoft Foundation Class 2.0 source code that programmers can extend or modify.

Figure 3. AppWizard Options Dialog

Step 3: App Studio

Once the initial program files are created, you can use App Studio to edit all the resources associated with the project. Accessed from the Visual Workbench Tools menu, App Studio is a Windows-hosted program that provides a graphical and interactive approach to designing user interfaces and editing resources of Windows applications.
App Studio includes editors for:

- Application menus
- Dialogs and forms
- Bitmaps, cursors and icons
- String table entries
- Accelerator tables

Editing your application's user interface in App Studio is similar to drawing your interface in the Visual Basic programming system. In addition to letting you draw menus and dialogs, App studio also supports drawing bitmaps, icons and cursors. App Studio lets you open multiple resource files and drag and drop resources between them. It is 100 percent compatible with the Microsoft Resource Compiler − allowing you to edit resources for both C and C++ applications. And it is fully integrated with Class Wizard and the Visual Workbench so that you can draw user interface

objects and connect them to code quickly and error free. In total, App Studio provides a completely integrated user interface editor to make drawing and coding user interfaces easy and efficient.

The following diagram shows a simple dialog being edited with the App Studio dialog editor:

Figure 4. App Studio

Step 4: ClassWizard

While working with resources in App Studio, the next step is to run ClassWizard, which you select from the App Studio toolbar. ClassWizard allows you to quickly connect the components of your program's user interface (menus, dialogs, etc.) to the code that handles them. ClassWizard is important because it allows you to browse and change messages, message-handler functions and member variables many times faster (and with fewer errors) than would otherwise be possible manually.

Once inside ClassWizard (see the following figure), you can select which messages from Windows the resource will respond to, and enter the names of functions that will handle the messages. ClassWizard updates the appropriate header files with all the necessary variable and function declarations. It also adds "empty" function definitions in the implementation files, along with comments to help you understand what code you need to add. At any time, you can jump directly to these functions in your source code by clicking the Edit Code button. You can also press F1 on a Windows message to display a Help screen on that message.

When relationships between a program's user interface and its code change, you can use ClassWizard to automatically edit the class members, message maps or dialog data macros within the source files to keep them consistent. However, ClassWizard is completely noninvasive; it never changes any code you write and is therefore completely safe. ClassWizard only does exactly what you tell it to do.

Class Wizard also allows you to set up and manage the new dynamic dialog data exchange and validation (DDX/DDV) capabilities of the application framework. DDX/DDV is a capability that allows data values entered into an application dialog to be automatically validated and assigned to member variables at run time.

Figure 5. ClassWizard

Step 5: Implementation with Microsoft Foundation Class 2.0

At this point, a complete skeleton application with all the appropriate resources and declarations has been built interactively and without any programming. The next step is to use the integrated Visual Workbench editor to complete the function implementations that respond to user interaction and add any other code necessary to complete the application.

To do this, you can use the Microsoft Foundation Class 2.0 library, an application framework built on the core set of functionality in Microsoft Foundation Class 1.0. Microsoft Foundation Class 2.0 provides an architecture and a set of prebuilt components that significantly simplify programming for Windows in C++. It offers a high level of abstraction that lets you focus on the details specific to your application, while simultaneously providing access to lower-level classes and the native Windows API for maximum flexibility and power.

Microsoft Foundation Class 2.0 offers more than one hundred reusable and extensible C++ classes, many of which already encapsulate the basic building blocks of a professional application written for the Windows environment. For example, to add a toolbar to an application requires fewer than 10 lines of code, but this represents more than 1,500 lines of Microsoft Foundation Class 2.0 framework code. Similarly, adding printing and print preview capability can be done in just a few minutes, but represents several thousand lines of framework code.

Because Microsoft Foundation Class 2.0 code can be intermixed with native Win32 or Win32s SDK calls, programmers can migrate existing 32-bit Windows-based applications to C++ at their own speed. Additionally, the Microsoft Foundation Class 2.0 code is completely portable between 16-bit and 32-bit Windows.

Step 6: Build and Debug With Visual Workbench

After the appropriate functions have been implemented, you can build your application by clicking the Build button on the Visual Workbench toolbar. The Visual C++ compiler offers all the size and speed benefits of the C/C++ 7.0 compiler, and adds additional features (see size and speed improvements section). As a result, Visual C++ produces executable files that are still among the fastest and smallest in the industry.

Visual Workbench can automatically create and maintain build options in its own makefiles, or you can use existing project files, which are 100-percent compatible with the NMAKE command-line tool from the Microsoft C/C++ development system version 7.0 and from the Win32pre-release SDK. You can use existing makefiles from C/C++ 7.0 or the Win32 pre-release SDK if you want to use special build rules or to invoke command-line tools.

You then debug your application using the Visual C++ debugger, a completely new Windows-hosted debugger that is fully integrated with the Visual Workbench.

Figure 6. Debugger Breakpoints Dialog

To set a breakpoint, select a line of source code in the editor and click the breakpoint button on the Visual Workbench toolbar. You can edit source code or set new breakpoints while you debug, or vice versa.

Key Points

The product overview previously discussed demonstrates the following key points about the design of Visual C++:

- **Delivers the benefits of Visual Basic to programmers using C++.** Visual C++ offers many of the ease-of-use and productivity benefits associated with the Visual Basic™ programming system while providing complete access to the power of C++ for programming in Windows.

- **Highly integrated tool set.** Visual C++ provides a highly integrated tool set that makes the development cycle more efficient from start to finish. This integration allows programmers to switch rapidly between distinct tasks such as editing, building and debugging, or to even do some of these tasks concurrently.

- **Full-featured applications with less work.** Through Microsoft Foundation Class 2.0, programmers can build better applications in a fraction of the time that it takes using C and the Win32 or Win32s SDKs. This is because the Microsoft Foundation Class Library 2.0 contains a large amount of reusable code, high-level abstractions and "canned" functionality that programmers can add to their applications with very little effort.

- **Easy to learn.** Its graphical and visual nature helps make Visual C++ easy to learn. Since all of Visual C++ (including all the tools) are now 100-percent Windows NT-based, all of the advantages of the Windows NT environment (rich user interface graphics, consistent operation, multitasking, toolbars, floating palettes and so on) are offered to programmers. Comprehensive product testing at Microsoft's usability lab helped to refine all aspects of the user interface and make it both fast and intuitive. New with Visual C++ 32-Bit edition is the extensive on-line documentation: over 10,000 pages of documentation is included in browsable, cross-referenced, and hot-linked **Books Online** form. Printed documentation is also available. Extensive sample code, online help and product support are available to assist developers of all levels of experience.

- **Easy migration path.** Microsoft Foundation Class 2.0 allows programmers to easily migrate their existing C/Win32 SDK program code to C++. Developers can freely mix Microsoft Foundation Class 2.0 code with native Win32 SDK or Win32s SDK calls, enabling programs to be gradually transitioned into C++ instead of rewriting them from scratch. Also, since the Microsoft Foundation Class 2.0 library follows function and parameter naming conventions that closely match the Win32 SDK, a developer's knowledge of programming for Windows NT is preserved.

- **Still the smallest and fastest executables files.** Visual C++ incorporates Microsoft code optimization technology that is now in its fifth generation. All of the optimizations of C/C++ 7.0 and in the Win32 pre-release SDK compiler are available in Visual C++, and a number of further improvements have been made, which are described below. In short, executable files built with Visual C++ 32-Bit Edition are still the smallest and fastest in the industry, and are smaller and faster than those built with any previous

Microsoft compiler.

The remainder of this guide discusses additional improvements and features in greater detail.

Executable Size and Speed Improvements

All of the code optimizations of C/C++ 7.0 and the Win32 pre-release SDK compiler are available in Visual C++ along with the additional improvements described below. As a result, executable files built with Visual C++ are still the smallest and fastest in the industry, and are even smaller and faster than those built using C/C++ 7.0 - the compiler that set the standard for small and fast code.

Smart Linking

Smart linking is a capability that can significantly reduce executable size by eliminating functions that are not referenced in the program. It is especially important for programs that use large code libraries (such as C++ class libraries) because linking with libraries usually results in the incorporation of a considerable number of unused functions.

New optimizations for Pentium and i486 Processors

The compiler generates code specifically for the 80386, 80486, or Pentium instruction sets, which results in faster execution and smaller code. Visual C++ 32-Bit Edition includes new optimization specifically for Pentium and i486 processors. These optimizations take advantage of the superscalar architecture of the Pentium processor to deliver even faster application performance than ever before.

Simpler Compiler Switches

A number of programmers asked Microsoft to incorporate two additional compiler switches, and we have done so: /O1 and /O2.
The /O1 switch provides all the best optimizations for size, while the /O2 switch provides all the best optimizations for speed. These further simplify the selection of build options.

Another improvement is the simplification of the process to create and use precompiled headers. Now, by specifying the /YX switch, a precompiled header is automatically created if none existed, and the file is automatically reused in subsequent builds.

32-bit Programming

Microsoft Visual C++ 32-bit Edition is designed to write 32-bit applications for either Windows 3.1 or Windows NT. 32-Bit applications offer higher performance, and because of the flat 32-bit address space, are often easier to program. When writing 32-bit applications, developers need not worry about various compiler memory models and segment sizes as they have had to with traditional 16-bit development systems.

Developers write 32-bit applications using the Win32 or Win32s API. The Win32 API is the complete 32-bit API for the Windows NT operating system. It includes support for preemptive multitasking, symmetric multiprocessing, bezier curves, security and more. The Win32s API is a subset of the Win32 API that allows applications to run binary-compatibly on Microsoft Windows 3.1. It is an operating system extension that allows Windows 3.1 to run Win32-based applications.

Win32s offers software developers the following:

- A powerful 32-bit programming model for Windows 3.1, which is binary compatible with Windows NT
- The performance advantages of 32-bit mode
- Win32 semantics for the API
- A rich subset of the full Win32 API
- The ability to ship Win32 applications today for Windows 3.1 and Windows NT

Win32s consists of a virtual device driver and a set of dynamic link libraries that extend Windows 3.1 to support Win32 applications. The Win32s files must be shipped with the Win32 application and installed on the Windows 3.1 system. Visual C++ 32-bit edition provides the Win32s binaries and a setup program to install Win32s on a Windows 3.l system.

Long-term Investment

Microsoft strives to make its family of language products a safe choice for long-term investment. The Microsoft Foundation Classes have become widely used among ISVs and corporate developers, and Microsoft has publicly stated its long-term commitment to MFC as the future direction for Windows-based and Windows NT-based programming in C++.

Microsoft Foundation Class 2.0 extends the Microsoft Foundation Class C 1.0 library so that the investment in Microsoft Foundation Class 1.0 applications is preserved. Microsoft Foundation Class 1.0 programs need only be recompiled to begin taking advantage of Microsoft Foundation Class 2.0. Future versions of Microsoft Foundation Class will offer the same backward compatibility so that programs written to previous Microsoft Foundation Class library versions will be able to run without a problem. Programmers can therefore incorporate new capabilities of new Microsoft Foundation Class libraries at their own pace.

Microsoft Foundation Class is also designed for compatibility with new versions of the Windows operating system. Applications written with the Microsoft Foundation Class 2.0 library on Windows version 3.1 can be recompiled and run with no source code changes on Windows NT operating system and Win32s API using Visual C++ 32-Bit edition.

Support

Microsoft provides a wide range of support options for developers, including our product support services, Microsoft Developer Network and CompuServe® forums. This extensive support network helps ensure that developers will be their most productive while using Visual C++. See Appendix A for more information on support services available from Microsoft.

Appendix A: List of Features

Visual Workbench

The Microsoft Visual C++ development system contains a completely new Windows-hosted development environment called Visual Workbench. Visual Workbench replaces the Programmer's Workbench of previous Microsoft C versions. Using Visual Workbench, programmers can edit, build, debug and browse C/C++ code from a single integrated Windows-based environment. Visual Workbench also offers seamless access to AppWizard for skeleton application generation, App Studio for creation and management of application resources and ClassWizard for connecting code to resources.

Features

- **Editor.** Windows-hosted. By default, language keywords, identifiers, comments and strings are displayed in different colors. You can disable the colors entirely or customize them to suit your own taste. Visual C++ 32-Bit Edition includes a new Find-in-files tool that allows global search through source files with many possible options.

- **Debugger.** Fully Windows-hosted. You can start the debugger from within the Visual Workbench and use toolbar buttons to set or remove breakpoints and step through the application source code. Visual C++ 32-Bit Edition includes support for Windows NT exceptions and threads, as well as a new memory window.

- **Class browser.** Windows-hosted, and can be used for both C and C++ code. You can now see graphical trees of class hierarchies and expand and collapse graphs by clicking a mouse on icons that represent class "nodes." This new method enables programmers to browse the class hierarchy more quickly than before. The browser can also display the relationships between calling and called functions.

- **Improved ease of use.** Throughout its development, the Visual Workbench designers refined the product's user interface by studying data from Microsoft's usability lab. As a result, the Visual Workbench interface is intuitive and designed to minimize the number of steps needed to perform each task, from editing through debugging.

- **Seamless access to all tools.** All tools, including App Studio, AppWizard, ClassWizard, the debugger, browser and editor can be accessed directly from Visual Workbench.

- **Multiple document interface.** Allows you to simultaneously display the debugger, source code windows, class browser information and status/error information.

- **Toolbar.** One-click access to frequently used operations for file saving, text finding/replacing, building, running and debugging.

- **Concurrent editing while building.** Allows programmers to edit source code while they build. For example, while a build occurs, a programmer can code a new function that will be compiled and linked during the next build.

- **External makefile support.** Visual Workbench enables programmers to run existing makefiles (called *external makefiles*) from within the Visual Workbench environment. In these cases, Visual Workbench does not alter the contents of the makefile, but allows programmers to maintain the contents of the makefile outside of the development environment. If the external makefile generates output compatible with CodeView®, the program can also be debugged from within Visual Workbench.

App Studio

App Studio is a major new component of Visual C++. It is a graphical, visual and interactive program that encompasses the following three programming tasks:

- User interface design and editing
- Resource creation, editing and maintenance

App Studio uses a completely graphical approach to user interface design, and resource editing. Despite the visual approach to designing resources, all of the control normally provided through the direct editing of resource files is still available in App Studio. If desired, programmers can always use the traditional method of directly editing resource files in the usual manner.

Features

- **Suite of tools.** App Studio includes a dialog editor, graphics (bitmap, cursor, icon) editor, symbol editor, string table editor, menu editor and accelerator table editor.
- **Completely integrated.** App Studio is fully integrated with both Visual Workbench and Microsoft Foundation Class 2.0 (through ClassWizard). At any time, programmers editing code within Visual Workbench can start App Studio by selecting it from the Visual Workbench menu bar. Control is returned automatically to Visual Workbench when resource editing is completed.
- **Resource browsing.** Allows you to view all resources associated with the application, either in summary (list) form or in detail through any one of the resource editors.
- **Simultaneous resource editing.** You can edit multiple resources at once in different windows. You can also open multiple resource files at the same time and drag and drop resources to move or copy them between files.
- **Drag and drop.** App Studio supports a wide range of drag-and-drop operations between resources that are being edited.
- **Automatic symbol management.** The symbol editor of App Studio lets you name and rename resources with text names while numeric values are managed automatically for you. This makes managing symbols much faster and less error-prone.

- **Modeless property sheets.** You can enter exact values for any object properties (such as button style) by displaying that object's property sheet. Property sheets are modeless, and they dynamically update as you click on different items. This allows properties for all user interface elements to be set very quickly.

- **Edits multiple file types.** Programmers can directly edit a wide range of file types, including .RC, .BMP, .ICO and .CUR. Since it reads .RC files, it is compatible with the command-line resource compiler of C/C++ 7.0 and the Win32 pre-release SDK.

AppWizard

AppWizard is a program that can be run as a standalone utility from Program Manager or from Visual Workbench. Its purpose is to create the "skeleton" application source code files that give programmers a "jump start" into programming with Visual C++ and Microsoft Foundation Class 2.0.

Features

- **Creates Microsoft Foundation Class 2.0 applications.** Creates working applications that contain Microsoft Foundation Class 2.0 functionality, including printing and print preview, support for Visual Basic custom controls, documents and views, multiple document applications, OLE support, context-sensitive help and toolbar capabilities. For programmers building a new application with Visual C++, AppWizard is an indispensable tool for getting started with the new Microsoft Foundation Class 2.0 features, such as the new document/view architecture. And AppWizard creates very little code since the implementation for core application features is provided by the Microsoft Foundation Classes 2.0 class library.

- **Creates all project files.** AppWizard creates not just the source files, but all the project files necessary to build the application immediately.

ClassWizard

ClassWizard is an important new tool that is the first of its kind. It can best be described as a "programmer's assistant" that helps developers with many of the details of C++ programming in the Windows environment while enabling them to have full control over every line of their source code.

Programmers can use ClassWizard to connect user interface controls (such as dialog buttons) to the application code that handles them. This replaces the old method of tediously opening and closing different source files to directly edit message maps, class declarations and dialog data routines. As a result, using ClassWizard is far more productive and less error-prone than editing this type of code by hand, especially for large projects with dozens or hundreds of files.

Features

- **Manage messages.** Allows programmers to map messages from Windows to their function handlers.

- **Browse messages.** Allows programmers to quickly browse all the messages and handler functions associated with user interface objects from a single location.

This is much faster than opening and scrolling through multiple source files. Context-sensitive help on each message is available at any time.

- **Dialog data exchange and validation.** Programmers can take advantage of the dialog data exchange (DDX) and dialog data validation (DDV) capabilities of Microsoft Foundation Class 2.0. ClassWizard maintains the relationship between dialog controls and their member variables in the dialog's class. For more information on DDX and DDV, see the following section on Microsoft Foundation Class 2.0.

- **Jump to code.** Programmers can jump directly to the correct source file to create or edit the implementations of the message handler functions. This is done through seamless communication with the Visual Workbench editor.

- **Completely safe.** ClassWizard never changes any code written by programmers. Whenever ClassWizard makes a change in a source file, it is only to sections of code that are maintained by ClassWizard. Since ClassWizard doesn't parse code written by programmers, it is error-free and has none of the interpretation problems inherent in code generators.

- **Completely integrated.** ClassWizard can be accessed from both Visual Workbench and App Studio so you can edit class information at any time.

- **Imports existing projects.** In just a few steps, you can import existing Microsoft Foundation Class 1.0 projects to make them compatible with ClassWizard.

Microsoft Foundation Class Library Version 2.0

Microsoft C/C++ 7 included the first version of the Microsoft Foundation Class library (Microsoft Foundation Class 1.0), an extensive set of classes that encapsulated most of the basic functionality of the Windows 3.1 SDK while providing higher-level classes that simplified programming. This class architecture, coupled with the internal mechanisms that implement the classes is called the *framework*.

In Visual C++ 32-bit Edition, the framework has been greatly enhanced through the introduction of Microsoft Foundation Class version 2.0, which offers programmers a set of higher level classes that simplify Windows and Windows NT-based programming. Through these classes, programmers are able to quickly create high-level objects that contain pre-built application behavior. For example, the new CFormView class allows programmers to build form-processing applications by encapsulating the functionality of scrollable data entry forms. Creating new forms requires almost no programming. Normally, features such as this would consume a considerable amount of development time. Microsoft Foundation Class 2.0 applications written using Visual C++ 32-bit Edition can be compiled to run under either Windows 3.1, using the Win32s API, or Windows NT, using the Win32 API.

If desired, programmers can still continue to build any functionality they desire using standard Win32 or Win32s SDK calls. Microsoft Foundation Class 2.0 therefore gives programmers a means of hiding the details of Windows-based development or fully exposing

it, depending on their needs.

The following is a list of the major new classes and capabilities offered by Microsoft Foundation Class 2.0:

Features

- **Documents and views.** A *document* is an object with which the application user interacts during an editing session, and is typically created by the New or Open commands on the application's File menu. Users interact with the data contained in a document through a *view* object, which represents the client area of a frame window.

  With the CDocument class, Microsoft Foundation Class 2.0 automatically supports high-level commands such as File New, File Save, File Save As and File Open. Microsoft Foundation Class 2.0 does all the work of displaying a dialog to gather information from the user and managing the disk file.

  Each document in a running application is attached to one or more *views*, which control the graphical display of application data on the screen. Programmers will typically derive a class from the Microsoft Foundation Class 2.0 CView class and then implement the display code. Once you implement drawing code within a CView object, the framework will automatically provide device-independent printing and print preview within your application.

- **Toolbar and status bar classes.** The new CToolbar class lets programmers easily create toolbars that have a row of bitmapped buttons and optional separators.

  The new CStatusBar class enables programmers to easily incorporate status bar capabilities into their applications. These include multiple paned bars that separate information into groups and can be used to display almost any text such as help messages or key state indicators.

- **Form and edit views.** CFormView provides form window capabilities for applications that present a user interface based on a dialog template that contains controls. CEditView allows you to create views with text editing and scrolling capabilities, and also supports printing, find and replace, cut, copy, paste and undo, as well as standard File commands (Open, Save, Save As).

- **Scrolling and split-pane windows.** Derived from CView, the CScrollView class provides documents with built-in scrolling capability, and CSplitterWnd splits Windows into multiple panes.

- **OLE classes.** Three categories of classes are provided to support object linking and embedding (OLE): OLE client classes, OLE server classes and an OLE exception class. These classes greatly simplify the work needed to implement OLE capabilities in an application.

  Microsoft Foundation Class 2.0 OLE support is tightly integrated with the document/view and command architectures, so you can easily implement OLE capabilities within objects belonging to the CDocument and CView classes.

The COleClientDoc and COleServerDoc classes allow programmers to create document objects that support OLE client or server capabilities, while inheriting all the features of the CDocument class. The COleServerItem and COleClientItem classes allow you to directly manipulate OLE items and manage either the server or client side of an OLE connection.

Also included is the COleException class, which enables handling of failures that occur during OLE processing. This class can be used by both clients and servers.

- **Dialog data exchange and validation (DDX/DDV).** Dialog data exchange/validation is a capability that allows data values entered into a dialog by the application user to be automatically validated and assigned to a member variable.

  Data from the dialog's member variables can be used to automatically initialize the values of the dialog's control when the dialog is displayed. You no longer have to write code to assign initial values to the controls.

  Also, data entered into a dialog by a user can be automatically assigned directly to the dialog's member variables when the dialog exits. You no longer have to write a Get function to get the data and assign it to a variable.

  Validating the data input to a dialog control can also be handled automatically. A range of default validations (such as number of characters entered into an edit box) are now performed automatically, and you can add custom validations to build on the existing functionality.

- **Context-sensitive Help.** Microsoft Foundation Class 2.0 provides an architecture that makes it easy to incorporate Help support in applications. Help support includes a Help menu with the standard commands, and a mechanism that maps command or resource IDs to the various help contexts. Help contexts are easily created in Visual C++ because every time a user interface element is created in App Studio, a help context for that element is automatically created. When a user presses the F1 key, Microsoft Foundation Class 2.0 automatically processes the keystroke as a help request for the current command target.

- **Compiler-independent.** Microsoft Foundation Class 2.0 source code is designed to be compiler-independent, enabling Microsoft Foundation Class 2.0 applications to be compiled with other standard C++ compilers. Compiler vendors need only supply a simple version header file (and possibly a version source file), and a project file before Microsoft Foundation Class 2.0 applications can be compiled.

For more detailed information on Microsoft Foundation Class 2.0, see the white paper on this subject.

Compatibility of Visual C++ with Existing Tools

Features

- **SDK compatibility.** Visual Workbench and App Studio are designed to work with C++ programming as well as with C programming using the Win32 SDK, or the Win32s SDK. Programmers who want to program in C and work entirely within one of the SDKs may do so. The same benefits that Visual Workbench and App Studio offer C++ programmers are also available to C developers.

  If desired, programmers can mix and match Microsoft Foundation Class 2.0 code with native Win32 or Win32s SDK calls, which makes it easy to gradually migrate existing SDK-based code to Microsoft Foundation Class.

  Since ClassWizard and AppWizard are tightly integrated with Microsoft Foundation Class 2.0, these are not available to SDK programmers.

- **Microsoft Foundation Class 1.0 compatibility.** Projects created with Microsoft Foundation Class 1.0 can easily be converted into the new Visual C++ project format at any time. Any source code written with Microsoft Foundation Class 1.0 can be rebuilt within Visual C++ with no problems. To make Microsoft Foundation Class 1.0 code work with ClassWizard, programmers must add a small amount of code to their source files. This involves placing simple C++ comment delimiters around program statements that ClassWizard will maintain.

Spy++ Analysis Tool

Spy++ allows a developer to quickly discover a tremendous amount of information about his or her Windows-based application. The display of window messages shows decoded parameters as well as return values. Flexible “filters” allow precise control over the messages, windows and other information displayed.

Figure 7. SPY++

To see Spy++ in action, start it and look at the Window, Process and Thread Tree views. Clicking on a “plus” sign next to a window in the Window Tree view shows its child windows. Double-click on a window name (the green color means that it is a hot-link) and you can open a Window Details view of that window to see detailed information about it. Double-click on the displayed window class to open a Window Class Details view for the class. Go back to the Window Details view and notice that the process and thread for that window are shown. Double-click on each of these to open a Process Details and Thread Details view for them.

Finally, go back to the Window Tree view and find a window for one of your running applications, like Program Manager for instance. Hold the Shift key down and double-click on its window handle and you will open a Message Stream view for that window. Run the mouse over the Program Manager window to generate some messages that will be displayed complete with return values and decoded parameters shown. Use the View menu filter commands to specify the messages, windows to include and output format of the display for the message

stream view. If you are unsure of what a specific message is, just double-click on its name in the message stream view and you will get full help on the message and its parameters.

Technical Support for Developers

Microsoft Product Support Services (PSS) provides support to individuals and organizations through a variety of channels, which are described below.

- **Microsoft forums on CompuServe.** The MSLANG forum is provided via the Microsoft Developer Services Area on CompuServe. This forum provides easily accessible, high-quality developer information through interactive dialog between peer developers and Microsoft developer support engineers. Also offered are developer-specific Microsoft Knowledge Base articles, the Software Library and query menus. A "no charge" area for bug reports and product suggestions is provided.

- **Microsoft download service.** Customers can use their own modems and terminal emulation software to access application notes, printer drivers and technical notes. This service is for downloading only; sending and receiving messages is supported through CompuServe forums. The number for the download service is (206) 936-6735.

- **Microsoft FastTips.** An interactive, automated touch-tone system providing support at no additional charge, via toll lines. It provides fast answers to commonly asked questions, delivered to the customer by phone recording or fax. It also includes a library of technical notes that are available by fax or mail. FastTips is available seven days a week, 24 hours a day, including holidays. The FastTips number for Languages is (206) 635-4694.

- **Dedicated support numbers.** Direct lines to Microsoft product support personnel via toll lines. These support lines are designed to answer set-up, install and general usage questions, and to receive product suggestions and bug reports. Hours for these lines are Monday through Friday, 6:00 a.m. to 6:00 p.m., Pacific time, excluding holidays. The number for Microsoft C/C++ products is (206) 635-7007.

- **TDD/TT (Text Telephone).** This service enables hearing-impaired users to access the same phone support provided for all Microsoft products and support levels. Access requires a TT modem. Available Monday through Friday, 6:00 a.m. to 6:00 p.m., Pacific time, excluding holidays.

Professional support. This service is designed primarily for larger companies seeking priority technical assistance on Microsoft products. Direct contact with senior Microsoft technical engineers is provided, along with guaranteed 24-hour response time, a problem escalation process and other services.

Premier support. The highest level of support, also designed for larger companies. Premier support includes individualized technical consulting, a dedicated technical account manager, access to senior Microsoft technical engineers, a response time of less than four hours and other services.

For details on any of these services, call PSS sales at (800) 227-4679, extension 11700.

Product Target

Visual C++ 32-bit Edition is designed for experienced C++ programmers who want to create powerful 32-bit applications for either Windows 3.1 using the Win32s API, or Windows NT using the Win32 API. It is the best choice for programmers who want their applications to run on the next generation of 32-bit operating systems. It includes complete software, documentation and extensive sample code on CD-ROM. Printed documentation may also be purchased additionally.

The following table lists the major features of Visual C++ 32-bit Edition

Feature	32-bit Edition
Visual Workbench	•
App Studio, AppWizard, ClassWizard	•
Microsoft Foundation Class 2.0	•
Windows-based application development	•
DLL development for Windows	•
MFC and user guide samples	•

Optimizing compiler •
Complete Win32 and Win32s SDKs •
Develop 32-bit Applications for Windows 3.1 •
Develop Applications for Windows NT •
Source profiler •
Debugger support for Windows NT Exceptions •
Debugger support for threads •
New Spy++ enhanced spy utility •
Books Online •

Figure 8

Availability, Pricing and Requirements

The Visual C++ 32-bit Edition is scheduled to be available August 24, 1993 at a suggested retail price of $599. The system requirements are as follows:

- An IBM®-compatible personal computer running Windows NT 3.1
- An Intel® 80386 or higher processor
- 16MB of available RAM (20MB recommended)
- One CD-ROM drive supported by Windows NT
- A hard disk with enough disk space to install the options needed. 6MB of available storage space is the minimum requirement to run your installation from the CD-ROM. To install the full configuration of the 32-bit Edition, 80MB of free disk space is required.
- A VGA (or higher resolution) adapter and monitor
- A mouse or other pointing device supported by Windows NT

Appendix B: An Exercise with Visual C++

The exercise in this Appendix gives you a brief tour of some of the components of Microsoft Visual C++. The goal is to demonstrate how quickly you can get started programming for Windows with C++ and Microsoft Foundation Class 2.0. This is not a comprehensive exercise that demonstrates all the features of the product, but is intended to show how Visual C++ allows you to quickly build C++-based applications for the Windows environment.

To perform this exercise, you must first install the Visual C++ 32-bit Edition on your computer. See the end of Appendix A for a list of system requirements.

Step 1: Create a Skeleton Application

In Program Manager, click the Visual Workbench icon in the Visual C++ group. From the Visual Workbench Project menu, select AppWizard, then enter a project name (for this example use "COOL"). Enter a project directory if you wish to put the project in a different location. You can do this using the directory browser in the dialog.

Click the OK button. You'll be presented with a confirmation dialog that lists the major features of your new COOL application. Click the Create button, and a Visual C++ project will be created for you. This project contains the files and declarations common to every standard Microsoft Foundation Class 2.0 application.

Unlike a code generator or a CASE tool, the skeleton application relies on the reusable code in the Microsoft Foundation Class 2.0 application framework to do most of the work. We'll look at these files in a moment. At this point, you can view the various features for your application using the Options and Classes buttons. Feel free to browse them, but for this exercise we'll demonstrate the power of Microsoft Foundation Class 2.0 and Visual C++ without adjusting them.

Step 2: Compile the Application

Click the Build button on the Visual Workbench toolbar (the one with the two downward-pointing arrows). While this project is building (it should take about 90 seconds on a standard 486/33-class machine) you can look at the source files in your project. Since Visual C++ allows you to continue editing while building, you don't have to wait for the build to finish.

To view the project source files, click the Project File button (the leftmost button) on the toolbar. Open MAINFRM.H, which contains the declaration for the main window of the application. As you can see, this declaration is just a class derived from the built-in Microsoft Foundation Class 2.0 class CMDIFrameWnd. Also notice the syntax coloring supported by the Visual Workbench editor, which makes your code more readable.

As you scroll downward through the file, you will see that the declaration is commented and documented. The class contains two member variables: one for the application's toolbar and one for its status bar. These variables use the two Microsoft Foundation Class 2.0 classes CToolBar and CStatusBar.

Now open COOL.CPP, which implements the application initialization code. Scrolling downward through the file, look for the InitInstance function which initializes many features of the application. For example, the LoadStdProfileSettings call automatically loads the most recently used file list in the File menu. The OnFileNew call creates an empty document for users of the application. The important point to note here (and in the rest of this exercise) is

that there is very little user-written code in this application, despite the large amount of functionality. Most of the statements are simply class declarations and function stubs ¾ all of the reusable code is in the Microsoft Foundation Class 2.0 library.

Step 3: Execute COOL.EXE

From the Project menu, select the Execute command (or press Ctrl+F5) to run COOL.EXE. Although you've written no code, you have a complete Microsoft Foundation Class 2.0 C++ application for Windows that contains the following features:

- An MDI application (SDI is optional)
- A toolbar with buttons that map to the menu commands File New, File Open, File Save, Edit Cut, Edit Copy, Edit Paste, File Print, and Help About
- A status bar with menu command prompt strings for all commands, and indicators for Num Lock, Scroll Lock and Caps Lock keys
- File menu commands for New, Open, Close, Save, Save As, Print, Print Preview, Print Setup, recent file list, and Exit
- Standard dialog support for obtaining file name information from the user (through the Open and Save As commands on the File menu)
- Standard dialog support for print and print setup
- Print preview support, including Next/Prev Page, Two Page view, and Zoom In/Out
- Edit menu hooks for Undo, Cut, Copy, Paste
- View menu to enable or disable the toolbar and/or status bar
- Window menu commands New Window (that correctly names each window by appending a numeric index), Cascade, Tile, Arrange Icons, and an active window list

Some of these features require you to write a small amount of additional code before they are fully enabled. For example, select the File Open command and you'll see that it requires you to add a small amount of code to read the data. However, notice that the most recently used list in the File menu is updated, and that multiple MDI child windows are automatically displayed with the correct title as you open files. There is even a default icon for documents when you minimize child windows. All of this functionality is handled by Microsoft Foundation Class 2.0, without requiring you to write any code.

Similarly, the File Save (and Save As) commands create a new file for you, but don't write anything to disk until you add application-specific code (which is added to COOLDOC.CPP's Serialize member function). Also notice that the Edit menu commands (and the accompanying toolbar buttons) are disabled because no code has been written to handle the Clipboard. However, many of these functions are fairly simple to write, and you can have a fully functional basic application for Windows that supports all of the above features with just a small amount

additional work.

Now exit the COOL application (be sure to exit COOL by using the File Exit command).

Step 4: Edit COOL's Windows Resources

Open the project's resource file by choosing App Studio in the Tools menu. This automatically starts App Studio, the integrated resource editor, and loads the projects resource file (COOL.RC). App Studio is a full-featured resource editor that lets you edit all of the standard resources of Windows 3.1 cursors, icons, bitmaps, accelerator tables, string tables and dialogs.

Select the Icon resource type to display a list of all the icon resources in the COOL project, then double-click the IDR_COOLTYPE icon. This is the icon that is used for minimized MDI documents. The App Studio graphics editor is started, and you can edit the icon any way you like. You might try clicking on a contrasting color and use the Fill Tool (the "paint can") to change the background color.

Now return to Visual Workbench by pressing Alt+Tab, or click on the Visual Workbench window if it is visible.

Step 5: Add Some Drawing Code

Using the Project File list, open COOLVIEW.CPP, which contains the code for the application's "view." The view is the window that displays the visible portion of the current document.

Scroll down the COOLVIEW.CPP file until you see the OnDraw member function. This function is the abstract device-independent drawing routine used by the framework. Completing this code automatically gives your application support for printing, print preview and standard drawing.

Replace the "TODO" comment with the following lines:

```
CString str = "Hello World";
pDC->TextOut(20, 20, str);
```

CString is a dynamic string class in Microsoft Foundation Class 2.0. The parameters to TextOut are the coordinates to draw the string and the string itself.

Step 6: Use ClassWizard to Connect Messages to Code

Select the ClassWizard item on the Browse menu command (Ctrl+W). This launches ClassWizard, a tool that enables you to connect user interface elements and messages to code, create new C++ visual classes, and manage dialog data exchange and validation (DDX/DDV).

Select the CCoolView class from the Class Name dropdown list, then choose CCoolView from the Object IDs list (this contains all of the various visual objects you can connect).

Select WM_SETCURSOR from the Messages list, which contains all the messages to which you can connect the selected visual object. In this case, CCoolView is a standard window so you see all messages appropriate for a window. If you are not sure what a message does, select the message and press F1 to display context-sensitive help on the message handler in Microsoft Foundation Class 2.0.

Now click Add Function (or double-click WM_SETCURSOR). This creates a new message handler for the message. If you are familiar with Microsoft Foundation Class 1.0, this is

equivalent to automatically adding the function to the message map.

Now click Edit Code, and you are taken to the appropriate place in the implementation file to add code for this message. This is very similar to Visual Basic, because you have gone from a message (an event in Visual Basic) to the code that is specialized for your application.

In the source file, replace the "return" statement and TODO comment, with the following two lines:

```
SetCursor(LoadCursor(NULL, IDC_CROSS));
    return TRUE;
```

Now, whenever the cursor is positioned over one of our views it will appear as a cross hair.

Note how ClassWizard helped you manage your classes from a single location, which makes it easier to visualize your application's classes, the messages they respond to, and the code that handles the messages.

Step 7: Connect User Interface Elements to Code

Remember that the Edit menu commands were all disabled when you first ran the application because no user code was written to handle the commands, and a default implementation was not provided by the application framework (because the Clipboard is application-specific).

Press Ctrl+W to once again display ClassWizard. Notice that the CCoolView class is already selected, because ClassWizard remembered where you left off. For the object ID, select the ID_EDIT_COPY user interface element. This represents the command for both the menu and the toolbar button. Double-click the COMMAND message to handle the command. ClassWizard suggests a function name, which you can accept. Note that the UPDATE_COMMAND_UI is also sent by the application framework and can be used to dynamically update menu items and/or toolbar buttons (e.g., enable or disable), depending on the state of your application.

Click Edit Code, which takes you directly to the code for handling the message. However, you don't need to add any code, since the framework now knows that your application is handling the message, so it enables the menu command and toolbar button.

Step 8: Run the Application

Run the application using the Project Execute command (Ctrl+F5). You'll be asked if you want to rebuild the project, so click Yes. Notice that all the files are saved for you, even the resource script, so there's no chance of losing any information.

When the application is displayed, notice that the cursor is a cross hair when positioned over a child window. Minimize a child window to see the new icon.

Notice that the Edit Copy menu item and toolbar button are now enabled, and that the text "Hello World" is in your view. Select Print Preview, which allows you to preview you document before printing. (Depending on the printer you have selected for your machine, you might need to Zoom In on the page to see the text clearly.) If you like, you can print your document because printing is fully enabled. With almost no effort, full Print Preview and Printing are already implemented for your application, allowing you to focus on more application-specific code. The Microsoft Foundation Class 2.0 Print Preview even maps the system font to an appropriate font on your selected printer.

Step 9: Continue to Explore

By now, it should be evident how Visual C++ can speed development of applications for the Windows operating system by supplying much of the functionality for you through Microsoft Foundation Class 2.0, and by providing a set of integrated tools that do much of the work for you. Although you've written only a few lines of code, your application has a large amount of ready-to-run functionality.

If you want to continue exploring on your own, you can add a menu item or a dialog, and explore compiler options (such as RETAIL) to see that the retail version of this application is smaller than 90K, despite the large amount of functionality. For more information on how to do this, see the Visual C++ Class Libraries User's Guide.

#########