IBM VisualAge for C++ for Windows

S33H-5030-00

# Installation Guide and Product Overview

Version 3.5

IBM VisualAge for C++ for Windows

S33H-5030-00

# Installation Guide and Product Overview

Version 3.5

**Note!**

Before using this information and the product it supports, be sure to read the general information under "Notices" on page vii.

**First Edition (February 1996)**

This edition applies to Version 3.5 of IBM VisualAge for C++ for Windows (33H4979, 33H4980) and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

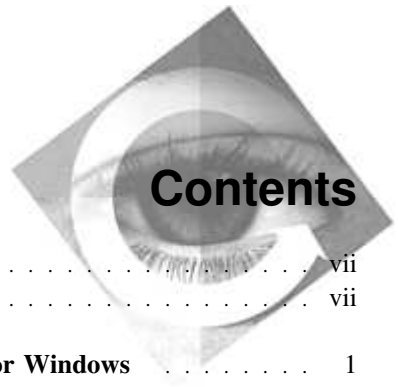
A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Canada Ltd. Laboratory
Information Development
2G/345/1150/TOR
1150 Eglinton Avenue East
North York, Ontario, Canada M3C 1H7

You can also send your comments by facsimile (attention: RCF Coordinator), or you can send your comments electronically to IBM. See "Communicating Your Comments to IBM" for a description of the methods. This page immediately precedes the Readers' Comment Form at the back of this publication.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

**© Copyright International Business Machines Corporation 1993, 1996. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.



# Contents


**Notices** . . . . . vii
Trademarks and Service Marks . . . . . vii

**Chapter 1. Introducing IBM VisualAge for C++ for Windows** . . . . . 1
About This Booklet . . . . . 2

**Chapter 2. Before You Begin** . . . . . 3
Hardware and Software Requirements . . . . . 3
Special Installation Considerations . . . . . 4
Preparing for LAN Installation (Optional) . . . . . 5

**Chapter 3. Installing VisualAge for C++** . . . . . 7
Choose an Installation Method . . . . . 7
Which Procedure Should I Choose? . . . . . 8
Choose a Target Directory for the Installation . . . . . 9
Install VisualAge for C++ (Condensed Instructions) . . . . . 9
Install VisualAge for C++ (Detailed Instructions) . . . . . 10
Install Win32s Support (Optional) . . . . . 15

**Chapter 4. Adding Components** . . . . . 17

**Chapter 5. Installation Utilities** . . . . . 21
Updating Your Desktop . . . . . 21
Deleting Components and VisualAge for C++ . . . . . 22

**Chapter 6. Now That You've Installed VisualAge for C++** . . . . . 25
What Do You Have? . . . . . 25
What Do You Do Next? . . . . . 27

**Chapter 7. If Something Goes Wrong** . . . . . 29
If the Component Requires Another Component . . . . . 29
If the Installation Program Fails While Copying Files . . . . . 29
If You Receive an Abort, Retry, Ignore, Go Window . . . . . 30
Network Neighborhood on Windows 95 . . . . . 30
The CD-ROM Installation Is Taking Too Long . . . . . 30
If You Receive an "Out of environment space" Message on Windows 95 . . . . 31
On Windows 95, You Have Run Out of Disk Space . . . . . 31
On Windows 95, the Operating System Will Not Come Up . . . . . 31
If a Tool Does Not Work or a DLL Is Not Found . . . . . 32
If You Have Problems with Header Files . . . . . 32


© Copyright IBM Corp. 1993, 1996

If Some Samples Will Not Run . . . 32
If You Can't Delete an Object Because It's in Use . . . 32
You Accidentally Deleted Your Desktop . . . 32
If Folders and Registry Updates Disappear . . . 33
If Bitmaps Are Discolored or Distorted . . . 33
If You Have Tried Everything and It Still Does Not Work . . . 33

**Chapter 8. VisualAge for C++ Product Overview** . . . 35
IBM VisualAge for C++ for Windows . . . 35
To Help You Get Started... . . . 35
Project Smarts . . . 35
WorkFrame . . . 35
To Help You Code... . . . 36
Visual Application Builder - An Object-Oriented Visual Application Builder . 36
Data Access Class Builder - Build Classes to Access Relational Data . . . 37
IBM Open Class Library - A Comprehensive Set of C++ Building Blocks . . 37
VisualAge Browser - Fast, Easy Access to Program Information . . . 41
VisualAge Editor - A Powerful, Language-Sensitive Editor . . . 42
Resource Workshop . . . 43
To Help You Generate Fast Applications... . . . 43
C/C++ Compiler - Generate Highly Optimized 32-bit code . . . 43
Direct-to-SOM (DTS) . . . 44
Performance Execution Trace Analyzer - Time and Tune Your Code . . . 45
To Help You Test and Debug... . . . 46
VisualAge Debugger - Find and Fix Coding Errors Fast . . . 46

**Chapter 9. VisualAge for C++ Online and Hardcopy Information** . . . 47
Online Information . . . 47
At a Glance . . . 48
Using VisualAge for C++ . . . 49
How Do I... Selections . . . 49
C/C++ . . . 49
Class Libraries . . . 49
Visual Programming . . . 50
IPF and Editing . . . 50
SOM . . . 50
Windows Programming . . . 50
Hardcopy Information . . . 51
PostScript Files . . . 52

**Glossary** . . . 53

**Bibliography** . . . 61

The IBM VisualAge for C++ Library . . . . . 61
C and C++ Related Publications . . . . . 61
Non-IBM Publications . . . . . 61

**Index** . . . . . 63

# Notices

Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent product, program or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY, 10594, USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independent created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Canada Ltd., Department 071, 1150 Eglinton Avenue East, North York, ONT Canada M3C 1H7. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

## Trademarks and Service Marks

The following terms are trademarks of the International Business Machines Corporation in the United States or other countries or both:

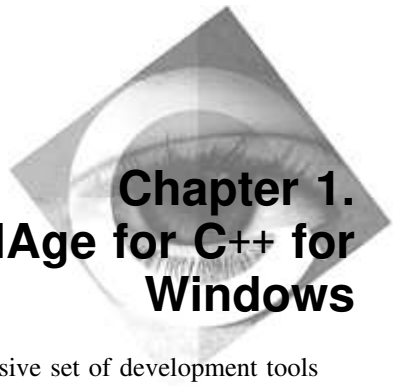
| | | |
|---|---|---|
| AIX | CUA | DB2 |
| IBM | IBMLink | MVS/ESA |
| OpenEdition | OS/2 | Presentation Manager |
| RMF | System Object Model | VisualAge |
| WorkFrame | Workplace Shell | |

Windows is a trademark of Microsoft Corporation.

Other company, product, or service names, which may be denoted by a double asterisk(**), may be trademarks or service marks of others.

IBM's VisualAge products and services are not associated with or sponsored by Visual Edge Software, Ltd..

© Copyright IBM Corp. 1993, 1996



# Chapter 1. Introducing IBM VisualAge for C++ for Windows

IBM VisualAge for C++ for Windows is a comprehensive set of development tools for creating 32-bit Windows applications in C and C++. VisualAge for C++ includes:

- WorkFrame, the customizable development environment that integrates the VisualAge for C++ tools and lets you plug in other IBM and non-IBM tools
- IBM's optimizing C/C++ compiler with support for templates, exception handling, structured exception handling, 64-bit integers, run time type information (RTTI), precompiled headers, and more
- An optimizing linker, featuring unreferenced code elimination for C and C++, as well as debug data compression
- An integrated library manager for both static and import libraries
- Visual Builder, a graphical tool for creating applications quickly and easily
- A programmable editor with syntax highlighting and other features
- A class browser to display your program elements and relationships in a graphical format
- A source-level debugger, with special features to make debugging C and C++ code easier
- A performance analyzer for tuning your program's performance
- The IBM Open Class Library:
  - Standard (I/O Stream and Complex mathematics)
  - Data Type and Exception (base, exception, string, date, and time classes)
  - Collection (generic container classes)
  - Database Access (for DB2 and other supported databases)
  - User Interface (classes that encapsulate the underlying presentation system to simplify how you develop Windows, OS/2, and Motif applications with graphical user interfaces)
- Toolkit, a set of tools for application development, including a resource compiler and make utility
- Sample programs and information to describe and demonstrate VisualAge for C++ features and coding techniques

© Copyright IBM Corp. 1993, 1996

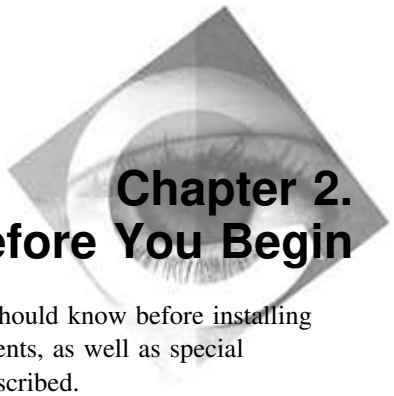
## About This Booklet

This booklet describes the VisualAge for C++ installation program, and gives you step-by-step instructions on how to install VisualAge for C++ from CD-ROM to a LAN or workstation or multiple workstations. It also suggests what you should do after you've completed the installation to familiarize yourself with VisualAge for C++ and get started using it.

This booklet also gives you a product overview of the VisualAge for C++ product.

Before using this booklet or VisualAge for C++ itself, you should be familiar with Windows 95, Windows NT, or Windows 3.1.

For additional information about IBM VisualAge for C++ for Windows, see Chapter 9, "VisualAge for C++ Online and Hardcopy Information" on page 47, which describes all the information that comes with VisualAge for C++ and tells you where to find it.



# Chapter 2. Before You Begin

This section contains important information that you should know before installing VisualAge for C++. Hardware and software requirements, as well as special considerations for certain types of installations, are described.

## Hardware and Software Requirements

**Processor**

32-bit processor (80386 minimum; 80486 or higher strongly recommended)

**Display**

VGA minimum
SVGA recommended

**CD-ROM**

Required

**Mouse or pointing device**

Required

**Operating System**

For the development environment, any of the following:

- Windows NT 3.51 or higher
- Windows 95.

For the execution environment, any of the following:

- Windows NT 3.51 or higher
- Windows 95
- Windows 3.11 with the Win32s V1.3 environment
- Windows 3.1 with the Win32s V1.3 environment.

**Other Software**

To use Data Access Class Builder, you need DB2 for Windows NT V2.1, Sybase SQL Server V10, or Oracle V7.

For remote debugging to Win32s, you need TCP/IP.

**Memory**

8M minimum for C development (12M recommended)
12M minimum for C++ development (16M recommended)
16M minimum for C++ Visual development (24M recommended)

**Note:** For Windows NT, add 4M to all memory requirements.

© Copyright IBM Corp. 1993, 1996

**Disk Space**

Disk space needs will vary. You can run VisualAge for C++ tool components directly from the CD-ROM, or you can install any or all of the tool components to your disk drive. If you choose to install all components to your disk drive you will require the following amount of space:

| | |
|---|---|
| All tools and toolkits | 285M |
| All samples and tutorial | 60M |
| All documentation | 25M |

You must also have at least 40M available for swap space.

**Time Requirements**

The following times are based on an installation done on a 90-MHz Pentium, 32 MB, 1 GB drive, double speed CD-ROM computer. Your installation times will depend on the type of hardware you are using, and the number of VisualAge for C++ components you are installing.

| | |
|---|---|
| CD-ROM | 1 minute |
| Minimal, to disk drive | 15 minutes |
| Complete, to disk drive | 40 minutes |
| Custom, to disk | Time will vary according to the type and number of components installed to disk. |

## Special Installation Considerations

There are several important points to be aware of before beginning an installation.

- If you have both Microsoft Visual C++ and VisualAge for C++ installed on the same computer, you may experience problems with header files. See Chapter 7, "If Something Goes Wrong" on page 29 for details before continuing.
- On Windows 95 only, your `autoexec.bat` will be modified by the installation program. A copy of your original, unmodified `autoexec.bat` file is saved as the next unused file name in the ascending sequence of `autoexec.001`, `autoexec.002`, `autoexec.003`, etc. A higher number in the file name extension indicates a more recent copy.
- On Windows NT only, the installation program sets VisualAge for C++ environment variables in the user environment for the user logged on at the time of the installation. It does not set these environment variables in the system environment table. Another user will not see the VisualAge for C++ product unless they too have the VisualAge for C++ environment variables set in their user environment.

  You can view and set user environment variables by selecting the SYSTEM icon in the Control Panel folder, which is accessed from the Main folder.

- You can install from either the VisualAge for C++ CD-ROM or from a LAN server. See "Preparing for LAN Installation (Optional)" on page 5 for more information about installing from a LAN server.

  You can also install VisualAge for C++ directly onto a LAN server.

- If you are reinstalling VisualAge for C++, you should first do the following:

  1. Delete the VisualAge for C++ installations from your system.
  2. Reboot your system.

## Preparing for LAN Installation (Optional)

If your workstations are connected to a LAN, you can set up the VisualAge for C++ installation files on a LAN server. Individual client workstations can then install VisualAge for C++ from the LAN server.

**Important!** When you install VisualAge for C++ across multiple workstations, make sure you observe the license agreement as described in the *License Information* booklet.

### If you are a LAN user:

Connect your workstation to the server containing the VisualAge for C++ installation files using the **net use** command:

```
net use x: \\machineid\netname
```

where:

- *x* is any free drive letter on your system
- *machineid* is the network name of the server
- *netname* is the directory resource on the server that contains the VisualAge for C++ installation files.

### If you are a LAN administrator:

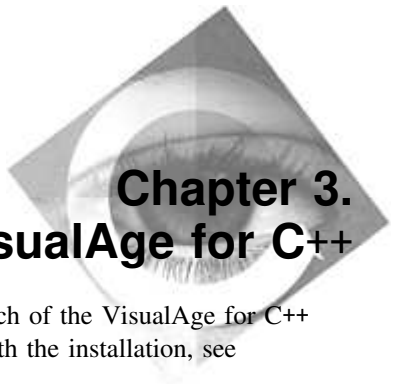
Follow the steps below to put the appropriate installation files onto the server:

1. Create a directory on the server that LAN users can access (for example, VACPP).
2. Change to the LAN directory.
3. Use the **xcopy** command to copy all the files from the CD-ROM to the server. If you `xcopy` the installation image to a server you should ensure the image's main directory is less than 30 characters. The syntax for **xcopy** is:

   ```
   xcopy d:\*  VACPP /s
   ```

   where *d* is the CD-ROM drive.

   **Note:** You do not need to copy the `ps` and `iocsrc` directories.
4. Notify LAN users of where the installation files are located.



# Chapter 3. Installing VisualAge for C++

This section gives you step-by-step instructions for each of the VisualAge for C++ installation procedures. If you encounter problems with the installation, see Chapter 7, “If Something Goes Wrong” on page 29.

## Choose an Installation Method

There are four basic types of VisualAge for C++ installation:

1. **Typical (complete) installation**

   This procedure installs all files for the components you select to your hard drive.

   Characteristics of a Typical installation include:

   - Fastest loading and running of tools.
   - Installation requires the largest amount of disk space.
   - Installation takes the longest amount of time.

2. **Minimal installation**

   For the components you select, this procedure installs a minimal number of files (executable and DLL) to your disk drive. The rest of the files remain on the CD-ROM or on a LAN server, and you access them from there as needed.

   Characteristics of a Minimal installation include:

   - Loading and running of tools will be slower than in a Typical installation, but faster than in a CD-ROM installation.
   - Installation requires less disk space than a Typical installation, but more than a CD-ROM installation.
   - Installation takes less time than a Typical installation, but more than a CD-ROM installation.

3. **CD-ROM installation**

   For the components you select, this procedure installs only those files that must be local to your hard drive. The rest of the files remain on the CD-ROM or on a LAN server, and you access them from there as needed.

   Characteristics of a CD-ROM installation include:

   - Loading and running of tools is slower than other types of installations.
   - Requires a very small amount of disk space.
   - Takes the shortest amount of time to install.

© Copyright IBM Corp. 1993, 1996

4. **Custom installation**

   This procedure lets you choose which components you want to install, and where each of those components is installed. You can always choose to add other components later.

   Characteristics of a Custom installation include:

   - Loading and running performance of tools may vary, depending on where you chose to install VisualAge for C++ components.
   - Disk space and installation time requirements will vary, depending on your choice of components and installation locations.
   - Certain components may require other components to also be installed.

Minimal, CD-ROM, and some Custom installations share the following characteristics:

- You cannot apply corrective service disks (CSDs) to the CD-ROM. If you want to apply fixes to VisualAge for C++ components, you must use a Typical installation.
- You cannot run all of the sample programs from the CD-ROM. Many of the VisualAge for C++ samples generate files, and because the CD-ROM is read-only, you won't be able to run them from the CD-ROM. You can copy the sample code onto your hard drive, or you can specify options for each sample to create all output files (including temporary files) onto your hard drive.

## Which Procedure Should I Choose?

Use Typical installation if:

- You want to maintain files locally on your hard drive
- You have enough room on your hard drive for the components you want
- You want to be able to apply corrective service to the product
- You want the tools to run as fast as possible

Use Minimal installation if:

- You want to minimize the disk space used on your hard drive, but still want reasonable performance

Use CD-ROM installation if:

- You want to minimize the disk space used on your hard drive
- You want to maintain common files on a LAN server for the team to use

Use Custom installation if:

- You want to be able to make your own decision as to whether to install the component on the hard drive or run it from the CD-ROM or LAN server.

## Choose a Target Directory for the Installation

Please keep the following considerations in mind when choosing a directory in which to install VisualAge for C++:

- We do not recommended installing to an operating system partition.
- We strongly recommend you install VisualAge for C++ to a new directory.
- The target directory's name must not be more than 30 characters in length.

## Install VisualAge for C++ (Condensed Instructions)

You can use the instructions in this section to install VisualAge for C++ if:

- you are familiar with installation procedures, and,
- you know where to find the VisualAge for C++ installation files (CD-ROM or LAN server directory), and,
- you have met all requirements specified up to this point in the document.

You can also use the detailed installation instructions described in "Install VisualAge for C++ (Detailed Instructions)" on page 10.

**Condensed Installation Procedure**

1. Verify disk integrity by running **scandisk** and/or **chkdsk** on both the operating system partition and the partition where you plan to install VisualAge for C++.
2. From the command line, change to the directory where the installation program is located (the root directory for CD-ROM; for the LAN server directory name, ask your LAN administrator).
3. To begin the installation, enter **setup** from the command line.
4. Choose the installation method you want: typical, minimal, CD-ROM, custom.
5. Choose the components and sub-components you want to install.
6. Specify the drive and directory where you want to install the components.
7. The **Start Copying Files** window appears, summarizing the selections you have made. Click on the **Next** push button to begin the installation, or click on the **Back** push button to revise your selections.
8. The **Setup Complete** window appears when the installation is complete.

   Before you can use VisualAge for C++, you must reboot your computer. Make your selection from the buttons in this window. Click on **Yes, I want to restart my computer now** or **No, I will restart my computer later**, then click on the **Finish** push button.

## Install VisualAge for C++ (Detailed Instructions)

This section includes step-by-step instructions to guide you through the VisualAge for C++ installation.

1. Verify disk integrity by running **scandisk** and/or **chkdsk** on both the operating system partition and the partition where you plan to install VisualAge for C++.
2. Insert the VisualAge for C++ CD-ROM or access the LAN where the VisualAge for C++ image resides.
3. From the command line, change to the directory where the installation program is located (the root directory for CD-ROM; for the LAN server directory name, ask your LAN administrator).
4. Read the README.TXT file, found in the same directory as the installation program. This file contains the latest information about VisualAge for C++ changes or restrictions, including those that affect installation. If instructions in the README.TXT file differ from this booklet, follow the README.TXT file.
5. To begin the installation, enter **setup** from the command line. The **Welcome to IBM VisualAge for C++ for Windows** window appears.
6. Click on the **Next** push button to continue with the installation. The **IBM VisualAge for C++ for Windows - Setup Options** window appears. See Figure 1.

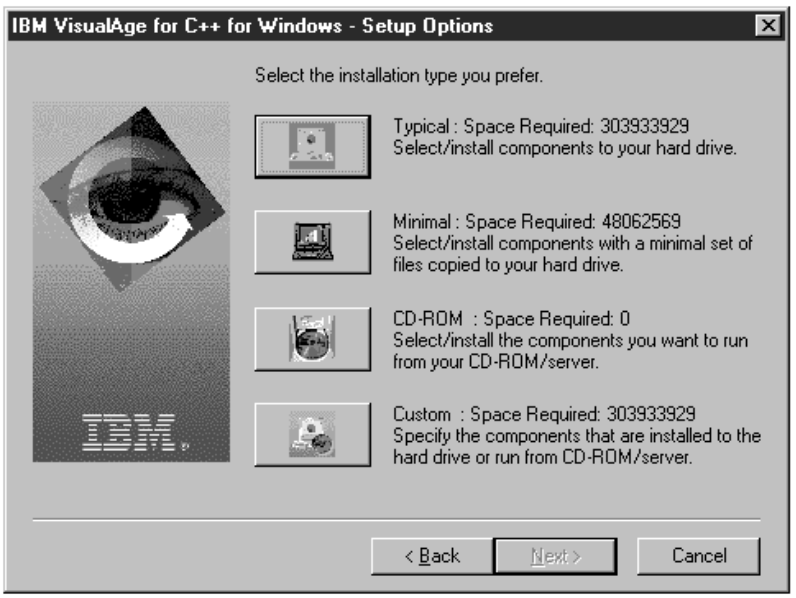



*Figure 1.*

7. The **IBM VisualAge for C++ for Windows - Setup Options** window shows the following installation methods:

   - Typical
   - Minimal
   - CD-ROM
   - Custom

   A `Space Required` line appears beside each installation method, showing the maximum amount of hard disk space needed to perform that type of installation. For an explanation on what each installation method does, see "Choose an Installation Method" on page 7.

   **Note:** Actual disk space requirements displayed may vary from those shown in Figure 1 on page 10.

   Select one of the installation methods. The **Select VisualAge for C++ for Windows Components** window appears. See Figure 2.

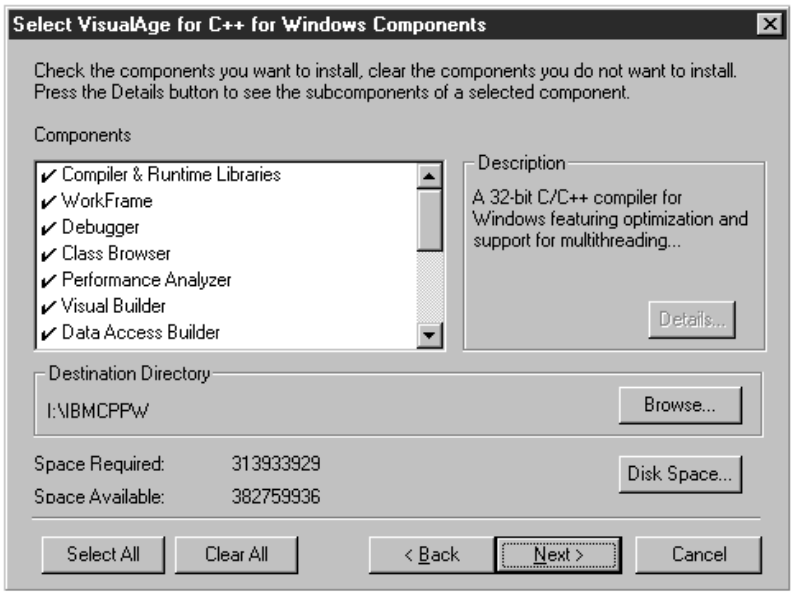



*Figure 2.*

8. The **Select VisualAge for C++ for Windows Components** window shows a list of components that you can install.

   Initially, all components are selected. You can select or deselect a component by clicking in the check-mark area to the left of the component name. You can also clear or set all selections by clicking on the **Clear All** or **Select All** push buttons.

Some components require that you also install other components for them to work correctly. If you select a component that has such a prerequisite, you will see a **Dependent Components Not Selected or Installed** window later in the installation, which lists the unfulfilled dependencies.

Some components have sub-components which may also be installed. To select the sub-components, click on **Details** (if this **Details** field is grayed out, then there are no sub-components to install for that component).

The **Space Required** field displays the maximum amount of disk space required to install your currently-chosen components and sub-components.

**Notes:**

a. You can find out more about a component by looking in the **Description** window to the right of the component list or, if you are doing a Custom installation, select **Description**.

b. For a Custom installation only, you can specify where you want each component installed. Choose one of:

- **Install to Hard Drive**, or,
- **Run from CD-ROM**.

If not specified otherwise, component installation defaults to the hard drive.

c. You do not have to install all the components at once. You can add components later by following the instructions given in Chapter 4, "Adding Components" on page 17. You must, however, select at least one component to continue with the installation program.

9. Specify the drive and directory where you want to install the components, keeping in mind the criteria specified in "Choose a Target Directory for the Installation" on page 9.

A default drive and directory is provided for you; everything is installed on your boot drive to a single directory called IBMCPPW. You can specify a different drive and/or directory with either of the following methods:

**Method One**

a. Click on the **Browse** push button.
b. Select a drive letter from the **Drives** field, near the bottom of the **Choose Directory** window.
c. Click on the **OK** push button.

**Note:** This is not the best method for choosing a drive, as it is primarily intended for choosing a directory.

**Method Two**

a. Click on the **Disk Space** push button to see how much space is available on your hard drive.

   The disk space window appears and shows you the space available on all drives, along with the space required for the components you have chosen.

   The installation program does not prevent you from installing to a drive that does not have enough space. If you do not have enough space, you will receive a notice, but will still have the choice to proceed or not to proceed.

   If you will be overwriting files, you may have enough space on a drive even though the installation program shows that you do not. If, however, there is not enough space, the installation program will fail. See Chapter 7, "If Something Goes Wrong" on page 29 for what to do if the program fails.

b. To change the drive, select the drive by clicking on it, then click on the **OK** push button. The installation program updates all of the directory fields to that drive, but does not change the directory name.

   To return to the main install window without making any changes, click on the **Cancel** push button.

c. Choose a directory:

   1) Click on the **Browse** push button.
   2) Select a drive letter from the **Drives** field, found near the bottom of the **Choose Directory** window.
   3) Use the **Directories** field to select your chosen directory.
   4) Click on the **OK** push button.
   5) Click on the **OK** push button again.

   You can also specify a new directory by manually typing its name in the **Path** field. If you specify a directory that does not exist, the system asks you to confirm that you want the directory created. Click on the **Yes** push button to confirm and create the new directory.

10. When you've chosen the components to install and specified the destination drive and directory, click on the **Next** push button.

    If in step 8 on page 11 you selected a component that has a prerequisite, but did not also select the prerequisite, the **Dependent Components Not Selected or Installed** window appears. See Figure 3 on page 14.

    Otherwise, the **Start Copying Files** window appears. Proceed to step 12 on page 14.

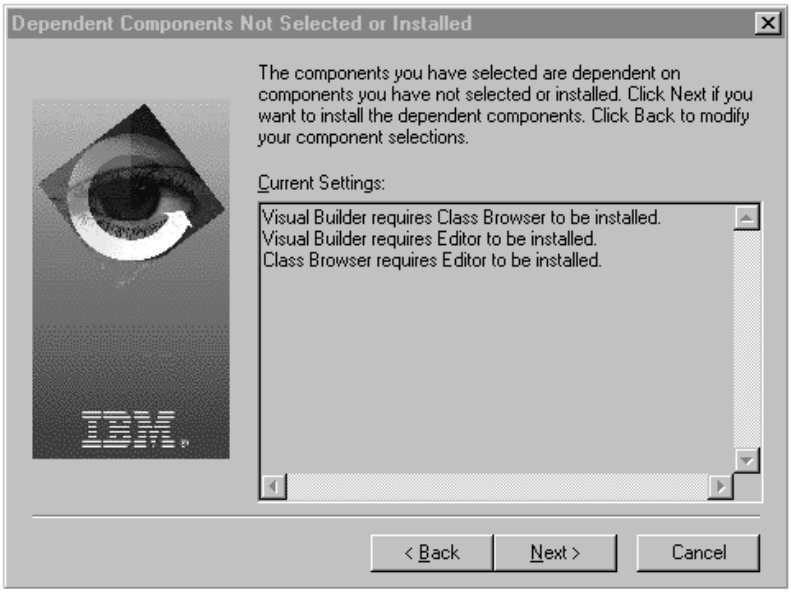



*Figure 3.*

11. The **Dependent Components Not Selected or Installed** window lists unfulfilled component prerequisites. To install these prerequisite components, click on the **Next** push button.

    **Note:** If you are doing a Custom installation, prerequisite components listed in this window will be installed to your disk drive unless you specify otherwise. You can change the installation location of a component by clicking on the **Back** push button, then selecting one of:

    - **Install to Hard Drive**, or,
    - **Run from CD-ROM**.

    If all prerequisite components are not installed, components that depend on those prerequisites may not work correctly when you try to use them.

    If you choose to install all components, then all dependencies will be met and you will not get this window.

12. The **Start Copying Files** window summarizes the selections you have made.

    If you are satisfied with these selections, click on the **Next** push button to begin the installation. The **Install Progress** window appears, and the installation program begins copying files to your disk drive.

    If you are not satisfied with your selections, click on the **Back** push button and revise your selections.

13. The **Install Progress** window displays the progress of the installation. The **Setup Complete** window will appear when the installation is complete.

    If you want to stop the installation program, click on the **Cancel** push button. A window appears asking you to confirm that you want to stop the installation. Click on one of the following push buttons:

    **Exit Setup** Close the window and end the installation program.
    **Resume** Continue installing the product.

14. Before you can use VisualAge for C++, you must reboot your computer. In the **Setup Complete** window, choose one of:

    - **Yes, I want to restart my computer now**, or,
    - **No, I will restart my computer later**.

    Click on the **Finish** push button to exit from the installation program.

You have now successfully installed VisualAge for C++, and are ready to go! See Chapter 6, "Now That You've Installed VisualAge for C++" on page 25 for what to do next.

## Install Win32s Support (Optional)

To run VisualAge for C++ applications on a machine using the Windows 3.1 or Windows 3.11 operating system, you must upgrade your system to include the Win32s 1.3 subsystem. Disk images for the Win32s subsystem are located in the following directory:

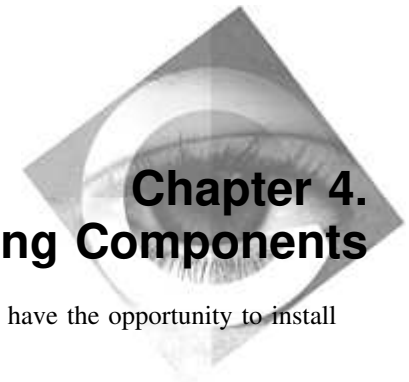
```
\IBMCPPW\SDK\DISKS\RETAIL\OLE32S
```

To install this subsystem on your Windows 3.1 or 3.11 machine, do the following:

- Run the **setup.exe** program in the DISK1 subdirectory, or,
- Create diskettes from these subdirectories, then run the **setup.exe** program on the diskette created from the DISK1 subdirectory.

To debug programs on a Windows 3.1 or Windows 3.11 machine, copy the `idebugp.exe` program from the `bin` directory on the VisualAge for C++ CD-ROM. You can find instructions for using the `idebugp.exe` program in the VisualAge for C++ *User's Guide*.

For information on developing VisualAge for C++ applications for the Win32s environment, see the IBM VisualAge for C++ for Windows *Programming Guide*.



# Chapter 4. Adding Components

Once VisualAge for C++ has been installed, you still have the opportunity to install additional components.

Use the installation program to add VisualAge for C++ components. The steps involved are as follows:

1. Insert the VisualAge for C++ CD-ROM or access the LAN where the VisualAge for C++ image resides.
2. From the command line, change to the directory where the installation program is located (the root directory for CD-ROM; for the LAN server directory name, ask your LAN administrator).
3. To begin the installation, enter **setup** from the command line. The **Welcome to IBM VisualAge for C++ for Windows** window appears.
4. Click on the **Next** push button to continue with the installation. The **IBM VisualAge for C++ for Windows - Setup Options** window appears. See Figure 1 on page 10.
5. The **IBM VisualAge for C++ for Windows Setup Options** window shows the following installation methods:

   - Typical
   - Minimal
   - CD-ROM
   - Custom

   Select one of the installation methods. The **Select VisualAge for C++ for Windows Components** window appears. See Figure 2 on page 11.
6. The **Select VisualAge for C++ for Windows Components** window shows a list of components that you can install.

   Initially, all components are selected. You can select or deselect a component by clicking in the check-mark area to the left of the component name. You can also clear or set all selections by clicking on the **Clear All** or **Select All** push buttons.

   Some components require that you also install other components for them to work correctly. If you select a component that has such a prerequisite, you will see a **Dependent Components Not Selected or Installed** window later in the installation, which lists the unfulfilled dependencies.

© Copyright IBM Corp. 1993, 1996

Some components have sub-components which may also be installed. To select the sub-components, click on **Details** (if this **Details** field is grayed out, then there are no sub-components to install for that component).

The **Space Required** field displays the maximum amount of disk space required to install your currently-chosen components and sub-components.

**Notes:**

a. You can find out more about a component by looking in the **Description** window to the right of the component list or, if you are doing a Custom installation, select **Description**.

b. For a Custom installation only, you can specify where you want each component installed. Choose:

- **Install to Hard Drive**, or,
- **Run from CD-ROM**.

If not specified otherwise, component installation defaults to the hard drive.

c. You must select at least one component to continue with the installation program.

7. When you have chosen the components you want to install, click on the **Next** push button.

   If in step 6 on page 17 you selected a component that has a prerequisite, but did not also select the prerequisite, the **Dependent Components Not Selected or Installed** window appears. See Figure 3 on page 14.

   Otherwise, the **Start Copying Files** window appears. Proceed to step 9 on page 19.

8. The **Dependent Components Not Selected or Installed** window lists unfulfilled component prerequisites. To install these prerequisite components, click on the **Next** push button.

   **Note:** If you are doing a Custom installation, prerequisite components listed in this window will be installed to your disk drive unless you specify otherwise. You can change the installation location of a component by clicking on the **Back** push button, then selecting one of:

   - **Install to Hard Drive**, or,
   - **Run from CD-ROM**.

   If all prerequisite components are not installed, components that depend on those prerequisites may not work correctly when you try to use them.

   If you choose to install all components, then all dependencies will be met and you will not get this window.

9. The **Start Copying Files** window summarizes the selections you have made.

   If you are satisfied with these selections, click on the **Next** push button to begin the installation. The **Install Progress** window appears, and the installation program begins copying files to your disk drive.

   If you are not satisfied with your selections, click on the **Back** push button and revise your selections.

10. The **Install Progress** window displays the progress of the installation. The **Setup Complete** window will appear when the installation is complete.

    If you want to stop the installation program, click on the **Cancel** push button. A window appears asking you to confirm that you want to stop the installation. Click on one of the following push buttons:

    **Exit Setup** Close the window and end the installation program.
    **Resume** Continue installing the product.

11. Before you can use VisualAge for C++, you must reboot your computer. In the **Setup Complete** window, choose one of:

    - **Yes, I want to restart my computer now**, or,
    - **No, I will restart my computer later**.

    Click on the **Finish** push button to exit from the installation program.

You have now successfully added components.

# Chapter 5. Installation Utilities

To update your desktop, delete components and the entire IBM VisualAge for C++ for Windows product, use the Installation Utilities program as described in this section.

## Updating Your Desktop

If you erase or change the VisualAge for C++ desktop folders, you can rebuild them in one of two ways.

**Method One**

1. On a command line, type **CPPUTILS** and press enter. The **IBM VisualAge for C++ for Windows - Installation Utilities** window appears. See Figure 4.

*Figure 4.*

2. Press the **Rebuild the desktop** button in the **IBM VisualAge for C++ for Windows - Installation Utilities** window. You will see a message that the desktop has been successfully updated.

© Copyright IBM Corp. 1993, 1996

**Method Two**

This method assumes that you can still get to your desktop and can see the Installation Utilities icon. If not, use Method One.

1. Click on the **Installation Utilities** icon found in the main VisualAge for C++ window. The **IBM VisualAge for C++ for Windows - Installation Utilities** window appears. See Figure 4 on page 21.
2. Press the **Rebuild the desktop** button in the **IBM VisualAge for C++ for Windows - Installation Utilities** window. You will see a message that the desktop has been successfully updated.

## Deleting Components and VisualAge for C++

To delete or reinstall a component, you must first delete the **entire** VisualAge for C++ product.

**Note:** The VisualAge for C++ deletion program looks for and uses the user environment variables defined by the installation program. If you are deleting VisualAge for C++ immediately after installing it, you must first reboot your system to set those user environment variables.

To delete VisualAge for C++, do the following:

1. Click on the **Installation Utilities** icon found in the main VisualAge for C++ window. The **IBM VisualAge for C++ for Windows - Installation Utilities** window appears. See Figure 5 on page 23.
2. Click on the **Delete IBM VisualAge for C++ for Windows** push button in **IBM VisualAge for C++ for Windows - Installation Utilities** window. A confirmation window appears.
3. Click on the **Yes** push button to continue deleting the VisualAge for C++ product from your disk drive.

   The installation program begins deleting files from your target installation directory. Note that **any** file in the target installation directory, whether it be your own or one from the VisualAge for C++ product, is deleted.

   When complete, you will see a message that VisualAge for C++ has been successfully deleted.
4. Shut down and reboot your system.

*Figure 5.*

**Notes:**

1. If VisualAge for C++ files are locked or in use during product deletion, the deletion process will fail, and a message is displayed.

   To recover from a failed VisualAge for C++ product deletion, do the following:

   a. Shut down and reboot your system.
   b. On Windows NT, go to the File Manager icon in the main VisualAge for C++ window and manually delete the rest of the product.

      On Windows 95, use Windows Explorer, found under the programs group to manually delete the rest of the product.

2. If for any reason you do not have an **Installation Utilities** icon, you can delete VisualAge for C++ by typing **CPPUTILS** at a command line and pressing the Enter key. When the **IBM VisualAge for C++ for Windows - Installation Utilities** window appears, press the **Delete IBM VisualAge for C++ for Windows** button to start deleting the product files.

# Chapter 6. Now That You've Installed VisualAge for C++

After you've successfully installed VisualAge for C++, you probably want to see what you have and find out how to get started.

## What Do You Have?

You should see the VisualAge for C++ window on your desktop for Windows NT, or in your programs group for Windows 95. If you installed all VisualAge for C++ components, the product folder will look similar to that shown in Figure 6.

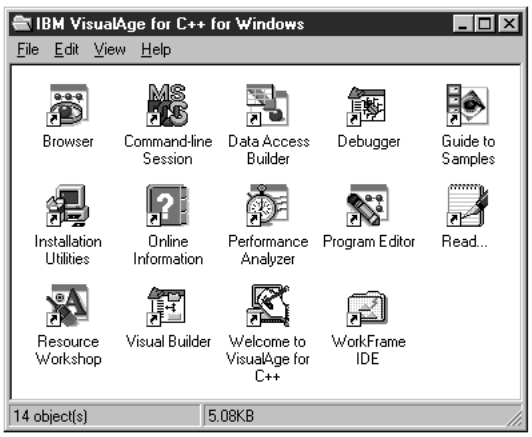



*Figure 6.*

Descriptions of each icon follow:

This is the README.TXT file that you read during the installation procedure. It contains information about any last-minute changes to VisualAge for C++ that could not be documented elsewhere. If the README.TXT conflicts with information in other books or files, please follow the README.TXT file.

This is an interactive tour of VisualAge for C++ that introduces the development environment and tools.

© Copyright IBM Corp. 1993, 1996

| | |
|---|---|
| Online Information | This icon contains the online information for VisualAge for C++, including user's guides and frequently asked questions. For a description of the online information included, see Chapter 9, “VisualAge for C++ Online and Hardcopy Information” on page 47. |
| Guide to Samples | This icon contains sample programs that demonstrate different programming techniques using VisualAge for C++. |
| Command-line Session | Select this icon to open a Windows command line which already has all the environment variables set for the VisualAge for C++ compiler. |
| WorkFrame IDE | Select this icon if you want a view of your project that puts all the tools and information in one window for easy and fast access. |
| Program Editor | Select this icon to use IBM's fully integrated, language-sensitive programmer's editor. |
| Debugger | Select this icon to help you find and fix coding errors. |
| Browser | Select this icon to display your program elements and relationships in a graphical format. |
| Performance Analyzer | Select this icon to time and tune your applications, analyze program hangs and deadlocks, view multithread, interactions and better understand your code. It gives you a view of your program's run-time behavior you just can't get through debuggers or browsers. |
| Visual Builder | Select this icon to rapidly prototype and build Windows applications complete with all standard Windows and OS/2 controls. In addition, the builder lets you use the advanced IBM Open Class Library extensions, such as canvases and graphic pushbuttons. |

| | |
|---|---|
| Data Access Builder | Select this icon to create new object-oriented database applications more quickly and reliably by having the builder generate the source code for you. |
| Installation Utilities | Select this icon to rebuild your desktop or delete VisualAge for C++. |
| Resource Workshop | Select this icon to create, modify, and compile windows resources for your application. |

If you installed VisualAge for C++ Win32 SDK Tools, you will see another window on your desktop for Windows NT or in your programs folder for Windows 95. For a description of the icons in this window, click on the **Online Information** icon in the main VisualAge for C++ window.

## What Do You Do Next?

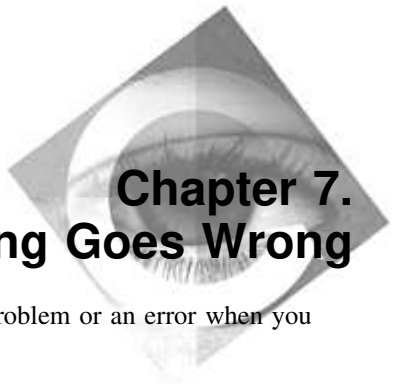
The online tutorial is an interactive tour of the product that introduces the WorkFrame development environment and the VisualAge for C++ tools using sample projects and a simple program example. We recommend you try this tour to familiarize yourself with using VisualAge for C++.

Help for tasks associated with the VisualAge for C++ tools is available from the **How Do I...** icon in the main VisualAge for C++ folder. The Debugger, Performance Analyzer, Browser, WorkFrame, Data Access Builder, Editor, and Visual Builder also provide this task help from their **Help** menus and contextual help panels.

The *Visual Builder User's Guide* contains a chapter that takes you through the development of a simple application using Visual Builder. The *Open Class Library User's Guide* provides a short Collection class tutorial and shows you how to rewrite a simple application using the User Interface classes.

You can get to all of this information from the **Help** menu of each tool. For more details on VisualAge for C++, see Chapter 9, “VisualAge for C++ Online and Hardcopy Information” on page 47.

You've installed the product, read the introductory material, and you're ready to go! We hope you enjoy using IBM VisualAge for C++ for Windows.



# Chapter 7. If Something Goes Wrong

This section tells you what to do if you encounter a problem or an error when you run the installation program.

## If the Component Requires Another Component

Many of the VisualAge for C++ components require that other components be installed for them to work correctly. If you select a component to install without selecting its prerequisite, you will get a **Dependent Components Not Selected or Installed** window listing the dependencies. Everything in the list will automatically be installed for you, so you can proceed by clicking on **Next**.

**Note:** If you are doing a custom installation, by default, all of the prerequisite components will be placed on your hard drive. By selecting **back** you can make your own installation choice; **Install to Hard Drive** or **Run from CD-ROM**.

## If the Installation Program Fails While Copying Files

If the installation program fails while it is copying files, take note of the error(s) you receive. Common errors are:

- The directory where your paging file or swap file resides is out of space.
- There is not enough space on your target disk.
- The target drive may be severely fragmented. In this case, you should run **scandisk**, **chkdsk**, or format and try again.
- A target file is readonly, hidden, or locked.
- The installation is receiving errors, timeouts, etc., while copying files from the CD-ROM or LAN server. This can be caused by:
  - the server being overloaded if you are installing from a LAN server.
  - a CD device driver problem if you are installing from a CD-ROM.
- The installation will automatically retry file copies. If you receive an abort, retry, ignore, or go dialog pressing **retry** usually bypasses this problem.
- If you are installing from a LAN server, you may not be authorized to access files and/or subdirectories that you need. If you are installing to a server, you may not be authorized to create files and/or subdirectories on the server. In either case see if you can manually copy files or create directories.
- A Windows full-screen exception or trap error occurs when performing the installation on Windows 95. Press the Enter key, and the installation should continue without error. In general, you should not have to press **Ctrl-Alt-Del** in this situation.

© Copyright IBM Corp. 1993, 1996

## If You Receive an Abort, Retry, Ignore, Go Window

If the installation program fails while:

1. Copying files
2. Updating the desktop
3. Updating the registry
4. Updating the `autoexec.bat` file
5. Updating Workframe sample projects

it will display a window giving you the following choices:

**Abort** This stops the installation.

**Retry** The action, if available, will be retried.

**Ignore** The last action you were trying to perform will be ignored and the installation will continue.

**Go** The installation will ignore any similar errors it encounters. For example, if a file copy fails and you select **Go**, the installation will continue and any further file copy errors the system encounters will be ignored.

## Network Neighborhood on Windows 95

Network Neighborhood on Windows 95 should not be used to call **setup** because your `autoexec.bat` file may get updated with the `\\server\alias` line in the environment variable settings rather than `x:\` (`x` being a partition name). VisualAge for C++ may not function under this condition. This will also lead to a very long path name in your `autoexec.bat` file.

Do not use Network Neighborhood on Windows 95 to call **setup**.

## The CD-ROM Installation Is Taking Too Long

When doing a CD-ROM installation or installing from a server you may find that it is taking an excessively long time and may even be giving you a number of file copy retry messages.

Write down the error message that is being displayed. You may also want to keep track of the number of retries the system is performing. The installation displays the retries in the **Install Progress** window.

Retries or failures can occur when the CD-ROM drive and/or device driver are not functioning properly or if a server is overloaded. Usually pressing the **Retry** push button in the error message window, once or as many times as prompted by the

system will correct this problem. If this does not work try to **xcopy** the files directly from the CD-ROM or server to determine if the same problems occur.

## If You Receive an "Out of environment space" Message on Windows 95

The "Out of environment space" message could occur on Windows 95 in one of two instances:

1. When the operating system runs the `autoexec.bat` file at bootup time
2. When running the `vacppenv.bat` command

In both instances try one of the following solutions:

1. Change the **CommandEnvSize** variable in the `system.ini` file, found in the `c:\windows` directory. This will increase the size of the environment variable space. How much you increase the size depends on what else is in the `autoexec.bat` file. For example, you could change the variable as follows:

   ```
   [NonWindowsApp]
   CommandEnvSize=10000
   ```

2. Create a `config.sys` file with the following statement:

   ```
   SHELL=C:\COMMAND.COM /p /e:15000
   ```

3. Go back to the original `autoexec.bat` file (the file called `autoexec.###`) and compare it with the new `autoexec.bat` file. Identify the changes made by the VisualAge for C++ installation and modify the `autoexec.bat` file manually, as required.

## On Windows 95, You Have Run Out of Disk Space

If you run out of disk space on Windows 95, use Windows Explorer from the programs group, to determine if there are any files on the target drive in the recycling bin. If there are, remove them to free up some disk space.

## On Windows 95, the Operating System Will Not Come Up

On Windows 95, if you reboot your system and the operating system comes up with an error stating that the `win.com` file cannot be found make sure you have a semicolon at the end of any path statements that appear before the VisualAge for C++ path statement in your `autoexec.bat` file.

## If a Tool Does Not Work or a DLL Is Not Found

If one of the following problems occur on Windows 95, then you may be out of environment space:

1. A tool you installed does not work when you try to use it.
2. You get an error that a DLL file is not found, but when checking the `autoexec.bat` file you find the path is set correctly.

Make sure all of the environment variable settings have been loaded into the operating system at boot up, by typing **set** in an MS-DOS prompt window.

## If You Have Problems with Header Files

If you have both Microsoft Visual C++ and VisualAge for C++ installed on the same computer and are getting Microsoft header files instead of VisualAge for C++ header files manipulate your **Include** environment variable to put the VisualAge for C++ headers first or remove the Microsoft headers entirely.

## If Some Samples Will Not Run

By default, sample programs are compiled with the Performance Analyzer installed and the /Gh switch on. If you have not installed the Performance Analyzer, your samples may not run.

## If You Can't Delete an Object Because It's in Use

When you delete VisualAge for C++, if any objects or files are in use, they are not deleted. A message informs you when this happens and instructs you to remove them manually. Shutdown and reboot your system. On Windows NT, go to the File Manager icon in the main VisualAge for C++ window and manually delete the rest of the product. On Windows 95, use Windows Explorer, found in the programs group.

## You Accidentally Deleted Your Desktop

If you accidentally deleted your desktop, go to a command line and type **CPPUTILS**. The **IBM VisualAge for C++ for Windows - Installation Utilities** panel will appear and from here you press the `Rebuild the desktop` button.

You receive a message that the desktop has been successfully updated.

## If Folders and Registry Updates Disappear

If you are installing under Windows NT with a userid that has guest access, registry updates and folders created during the installation may not remain after logging off.

Install VisualAge for C++ from a userid that does not have guest access.

## If Bitmaps Are Discolored or Distorted

After entering **setup** to begin the installation, you may find the bitmaps being displayed are discolored or distorted. This may indicate that other applications were running when the **setup** command was issued.

This will not have an affect on the success or failure of the installation. However, if you would like to display the bitmaps correctly, you may exit the installation program and reissue the **setup** command. To exit the installation program, click on **Cancel** in the **Install Progress** window. A window appears asking you to confirm you want to stop the installation; select **Exit Setup** to close the window and end the installation program. To continue installing the program, select **Resume**.

## If You Have Tried Everything and It Still Does Not Work

If the installation program continues to fail after you have tried everything suggested in this section, contact VisualAge for C++ Service and Support.

Make sure you have the following information available, to give them when they ask:

1. The type of computer you are using
2. The type of processor
3. How much RAM you have
4. The amount of hard disk space you have on the target drive and any operating system partition
5. The settings in your `autoexec.bat` file, or if this is unavailable, the settings in the standard `vacppenv.bat` file that is created when you install VisualAge for C++
6. A list of the components you have installed
7. Whether you installed from a CD-ROM or a LAN server
8. The operating system you are using: Windows NT or Windows 95
9. Any error messages you may have received
10. Any other information you believe to be important

You can contact Service and Support by:

| | |
|---|---|
| **Compuserve** | GO VISUAL |
| **IBMLink/ServiceLink/TalkLink** | For information on subscribing to these services, please call: |
| | 1-800-547-1283 (USA) |
| | 1-800-465-7999 ext. 228 (CANADA) |
| **Internet** | vacppwin@vnet.ibm.com |
| | http://www.software.ibm.com/ad/visualage_c++ |
| **Telephone** | 1-800-992-4777 in North America<br>Outside North America, contact your local IBM branch. |

For details on all the VisualAge for C++ Service and Support offerings, see the card *HELP!! and how to get it* included with the IBM VisualAge for C++ for Windows product.

# Chapter 8. VisualAge for C++ Product Overview

### IBM VisualAge for C++ for Windows

## To Help You Get Started...

Programmers have indicated they want simple and quick access to help when getting started. This release has been designed to provide you instant access to the power of VisualAge for C++ with:

- A main folder with all the tools and documentation of VisualAge for C++ for Windows.
- An overall product tutorial.
- An Information Notebook - Page tabs give you access to information grouped under User's Guides, References, and How Do I help.
- Sample Programs Notebook with accompanying documentation. You select a sample category (one of the notebook tabs). From this page you can open the README.TXT file for the sample and you can also open the sample within a WorkFrame project.

### Project Smarts

Project Smarts gives you a suite of projects which help you get started on your application as quickly as possible. You select the type of project you want to create and how you want to customize it. Project Smarts creates a complete boilerplate application which you can then customize to meet your needs. An example of a project type is a User Interface Class Library DLL. Press **Done**, after specifying some customization options and a project is created with a set of template files that will compile to build a working DLL. You don't need to learn which compile or link options are required, nor which libraries you need to include.

### WorkFrame

WorkFrame gives you a view of your project that puts all the tools and information in one window for easy and fast access. No matter what the complexity of the number of projects a developer has to deal with, WorkFrame provides the required access in an optimized way.

Highlights of WorkFrame are:

- Build Smarts provide a quick way to switch between the most common build options. Simply tell us if your code needs to be debugged, optimized, or

© Copyright IBM Corp. 1993, 1996

browsed, and leave the details of changing compiler and linker options to VisualAge for C++. If you want to deal with the full breadth of our compiler's flexibility, access to our compiler and link options, via dialog boxes, is still provided.

- There is a menubar and toolbar in each project window. For those more comfortable with direct manipulation, you can use popup menus to start actions or perform file operations.
- Intuitive and Capable Projects - tools launched from projects have access to all actions defined in the project. So when you pick source files to edit, you can ask for the project to be built right from the editor. When the project builds cleanly, you can start the debugger (or any other tool you like), also right from the editor.

## To Help You Code...

### Visual Application Builder - An Object-Oriented Visual Application Builder

With Visual Builder, you can rapidly prototype and build Windows applications complete with all standard OS/2 and Windows controls, such as menu bars, listboxes, etc, but in addition the builder lets you use the advanced IBM Open Class Library extensions, such as canvases and graphic pushbuttons. It is an advanced object-oriented visual application development environment.

The Visual Builder includes:

- A powerful visual editor that enables you to create complete applications, often without writing code
- An extensive library of prefabricated graphical user interface (GUI) parts plus program logic parts for application logic that are not part of the user interface
- C++ language support for creating new parts
- C++ code generator

Use Visual Builder to connect the graphical user interface to the logic and data of your application, then automatically generate the C++ source code. Prefabricated parts are supplied, but its capabilities are not limited to using just these parts. You can take advantage of the extensibility of Visual Application Builder by creating and adding your own reusable code to the parts palette. You can also import these parts from other applications or export these parts to other applications. The parts are actually treated as classes by Visual Builder. By developing a library of your own unique parts, tailored to the special requirements of your business, you can create even large and complex applications simply by visually arranging and connecting parts.

Visual Application Builder helps realize the advantages of OO programming, the ability to reduce programming time and improve code quality, by making it easy and practical to develop using a library of reusable components.

## Data Access Class Builder - Build Classes to Access Relational Data

Do your applications need to work with a relational database? Accessing data through C++ classes was once a labor-intensive and error-prone chore. The Data Access Class Builder allows you to create new object-oriented database applications more quickly and reliably by generating the source code for you. Add, update, delete, and retrieve methods are generated to manipulate data stored in DB2, Sybase or Oracle relational databases. Session control for multiple connections and transaction scoping is provided with the Data Access Class Library.

The generated database code can be used directly in your programs, or you can import them into the Visual Application Builder. By using Visual Builder to connect them to the GUI or other parts, you can quickly create high quality applications.

Some of the key features of Data Access Class Builder are:

- Map Tables to Classes - Create new classes using your existing database tables. You can create one class, or many classes, from any table. Both C++ and SOM IDL code are supported.
- Quick or custom mapping - The Data Access Class Builder offers a quick map feature that allows you to do column-to-attribute mapping. By using inheritance, you are able to customize your classes to suit your needs.
- Visual display of your mapping - The Data Access Class Builder graphically displays the mapping of your database tables to the object classes. This view allows visual editing and uses icons for tables and classes and relationship links to show the mappings.
- Connection and transaction services - A class library is provided for multiple connection and disconnection from your databases. In addition, commit and rollback operations are provided to handle transaction services.
- DB2 family support - Access DB2 in a stand-alone environment or access a remote DB2 through the DB2 Client Application Enabler and DDCS is supported.
- Sybase and Oracle access - Access for Sybase and Oracle relational databases is provided through the ODBC interface (new for the Windows version).

## IBM Open Class Library - A Comprehensive Set of C++ Building Blocks

The IBM Open Class Library provides you with a variety of building blocks to use in your C++ programs. Many of the fundamental components of C++ applications have been provided as reusable, extensible classes. Whether you are a novice programmer

or expert developer, the IBM Open Class Library can help you dramatically reduce your programming effort and avoid coding errors. That's because the IBM Open Class Library offers you a comprehensive set of classes: from basic input/output operations and string handling to multimedia and user interface support. The IBM Open Class Library brings you the true power of object-oriented programming technology.

Classes in the IBM Open Class Library support consistent programming interfaces across the entire range of platforms supported by VisualAge for C++: OS/2, AIX, MVS/ESA, OS/400, Sun Solaris** and Windows. This makes cross-platform porting faster, easier, and less error-prone. Following the documented guidelines, an application written to the core classes in Open Class can be easily ported to the other operating system environments. Use the powerful programming abstractions in the IBM Open Class Library rather than the low-level system-specific APIs. This can help reduce the amount of platform-specific code in your application.

Because the C/C++ language standard and class libraries were designed for portability, programs written for VisualAge for C++ for OS/2 can be recompiled with minimal system-specific coding required. The resulting application will have the "look and feel" of a native Windows application. Optionally, applications can have the IBM CUA look and feel through the use of style options.

These classes help build robust, reliable applications, having been tested and used extensively, including in IBM's own products.

The five parts of the IBM Open Class Library are:

- **Collection Classes** offers a complete set of abstract data types like sequences, sets, bags, trees, and so on. Some of the unique value the Collection Class delivers include:
  - Performance Tuning Facilities - - the classes are designed and implemented for optimum application performance. There are several underlying implementations to chose from, each optimized for different requirements: array, linked list, hash table, AVL tree and B*-tree. All provide the same programming interface; as you tune the performance of your application, you can select the best underlying implementation for the same data type abstraction, with minor source code changes.
  - Completeness of abstract data types (for instance, all abstract data types have both sorted and keyed version).
  - Consistent interface and functionality for all methods in all collections
  - Very flexible parameterization, (for instance, you can implement your own memory manager)

- Very flexible definition of element operations (default equality and comparison operators can be overridden with your own definition).

**Collection Class SmartGuide:**

VisualAge C++ comes with the Collection Class SmartGuide that helps you choose the appropriate Collection Class type for your application. The SmartGuide accomplishes this by asking you a series of questions about the properties of the class that you want to use, like use of keys and element equality.

Based on your selections, the SmartGuide chooses the proper class and generates code that includes the element type and key type.

The Collection Class SmartGuide can be invoked from the Editor menu bar each time you edit a **.cpp** or **.c** file.

For more information on how to use the Collection Class SmartGuide, see the Open Class Library User's Guide.

- **Compound Document Framework** within the IBM Open Class reduces the effort of developing OLE 2 containers and servers by generating most of the default behavior for the application developer. The framework is intended to provide a part architecture that will be consistent across both OLE 2 and OpenDoc. The Framework, which is the first injection of Taligent Technology into IBM Open Class, provides an abstraction from OLE 2 into higher-level classes. The framework reduces the coding of an OLE 2 component from a very large number of low-level OLE 2 API calls to a very small number of Compound Document Framework overrides. These include:
  - OLE 2 drag and drop
  - OLE 2 cut, copy, and paste
  - OLE 2 embedding
  - OLE 2 linking

  In other words, the framework generates most of the default behavior that an OLE 2 container or server needs. The framework also provides the basis for a portability layer between the OLE 2 component architecture and OpenDoc.

  Coding to the Framework also provides a model/view separation to the container or server, even without the use of OLE. This reduces maintenance of an application by isolating model (data) objects from view (presentation) objects.

- **User Interface Classes** provide a consistent set of classes for programming graphical user interfaces on OS/2, AIX, Solaris, and Windows. These classes simplify the coding of GUI applications and result in a more portable, reusable OO user interface. IBM Open Class Library gives you a consistent way to write user interface code without having to program to the low-level system APIs. The IBM Open Class Library exploits the underlying operating system services for

you; for example, the user interface classes exploit Motif** services on AIX and Presentation Manager services on OS/2. (There are some platform extensions in each environment that have been noted in the IBM Open Class Library documentation.)

Using the User Interface Class Library, you can easily and quickly:

- Create and display windows using title bars, varying sizes and styles, and more
- Include controls such as menus, buttons, text, list boxes, sliders, notebooks, and containers, with new support for animated pushbuttons
- Draw canvases with flexible window layouts that allow automatic redrawing and aligning of cells independent of the display device
- Provide direct manipulation (drag/drop) support
- Add multimedia with classes for constructing many different devices, like a MIDI sequencer, wave file playing and editing, digital video players/recorders, and programmable CD players
- Provide 2-D graphics support for drawing primitives (lines and arcs), plus support for reading and displaying various graphic formats, such as GIF, PIF, BMP, and others
- Create and manage a tool bar, including being able to move, change, and resize it
- Create parts, using the parts notification framework, for the new Visual Builder.
- Communicate between applications running on the same machine using DDE (Dynamic Data Exchange)
- Display help, define contextual help, and handle help keys, plus use new fly-over help that comes up automatically when the mouse pointer is on an object
- Cut, copy, and paste to and from the clipboard

You can use all of these classes as provided or extend and tailor them, and combine them with the other class libraries to easily develop complete object-oriented applications.

- **Application Support Classes** provide the basic abstractions needed during the creation of most C++ applications, the date and time classes provide you with data types to store and manipulate date and time information. The exception classes provide the framework for throwing exceptions within the class libraries. The trace class helps in tracing exceptions.
- **Standard Class Library** consists of the de facto standard I/O Stream library for C++ input and output, and the Complex library for manipulating complex (imaginary) numbers.

## VisualAge Browser - Fast, Easy Access to Program Information

The introduction of the OO paradigm in C++ has resulted in a marked shift in programming techniques and requirements. If you are a C programmer, you deal with large collections of modules, data types, or functions. As a C++ programmer, you now must deal with large and sometimes complex collections of interrelated classes in class libraries. The Browser helps you understand and use these class libraries with the following features:

- It displays the subtle nuances of C++ inheritance, including the complexities of multiple inheritance.
- It presents complex C++ programs and class libraries quickly, in an accurate, easily understood graphical format. Other class browsers might be equally fast, but they require you to recognize and filter invalid data.
- It allows you to browse C++ source code without needing to compile first, by using the built-in QuickBrowse feature. Other browsers require you to be able to either compile or even link your program or library before you can browse its contents.
- It uses the simple object-action paradigm. You select something, and the second mouse button gives you a list of actions available for the object that you have selected.
- A novel approach to displaying class members (using an IBM Open Class container object) allows you to focus in on the aspects of a class that you are interested in without introducing cumbersome interfaces such as filters, SQL (or worse, non-SQL) querying mechanisms.
- You can trace the static call graph of functions to understand who is calling whom, even in very complex applications.
- The web of file inclusions is displayed graphically. This helps you understand the complex nature of file inclusion that is frequently a result of programming with class libraries.
- You can determine what possible types of C++ exceptions can be thrown if a function is called. This can help you react to all possible error conditions that the function could encounter.

The Browser also comes with a full complement of "bells and whistles":

- You can customize the user interface: colors, line styles, fonts, double-click semantics, and the amount of assistance that you receive.
- You can print or save any image: flat files or WYSIWYG. You can create bitmap files and then use them in your program documentation.

- You can take advantage of multiple windows to see more than one thing at a time. For instance, you can keep a list of class names available, and also look at a particular class details.
- The Browser supports SOM objects, and is integrated with the debugger, editor, and compiler.

## VisualAge Editor - A Powerful, Language-Sensitive Editor

A powerful 32-bit editor has been integrated into VisualAge for C++'s suite of tools. This editor is fast, simple to use, helpful, and easily modified to meet your personal preferences.

You can compile, browse, make, build, debug, or issue other VisualAge for C++ commands without leaving the editor. It performs all the common editing tasks such as insert, delete, split and join lines, find, block and manipulate text, undo changes, create and find marks, and move between different source file views. However, it does much more to help make programming simple and error-free:

- It is language-sensitive, with highlighting in different fonts and colors for different types of language constructs. Automatic indenting and dynamic error checking is provided. You can correct your work by quickly navigating through each error without having to compile or leave the editor.
- The editor helps you to review and understand the code by presenting several views of the source. For example, display only function headers in the file to quickly see what is in the file and locate a function of interest. You can also display the program's flow of control, or insert programming templates.
- The editor is fully customizable. You can:
  - Change key assignments
  - Write external commands to extend editor capability
  - Create additional parsers
  - Configure the tool bar
  - Select a "personality" (It supports the look and feel of many common editors, such as BRIEF or EPM.)
- Finally, you can record a sequence of keyboard events as an editor macro command and then modify it to be reused as an external command.

### Resource Workshop

Create the Windows resources that you need for your application using the Resource Workshop. This utility provides an integrated environment of specialized editors for creating, modifying, and compiling the following resources for your Windows applications:

- Menus
- Dialog boxes
- Cursors
- Icons
- Version information
- Bitmaps
- String tables
- Accelerator keys
- Fonts
- User-defined data

If you are migrating your applications from OS/2, use the Resource Image Conversion Utility to convert your existing OS/2 icons and bitmaps to Windows format.

You have the choice of creating help which can be displayed by the Information Presentation Facility (IPF), or through the native Windows help management system.

## To Help You Generate Fast Applications...

### C/C++ Compiler - Generate Highly Optimized 32-bit code

Applications that perform well require tools that generate highly optimized code and make efficient use of disk and memory space. The VisualAge for C++ compiler and linker represent state-of-the-art technology; the applications you write will be fast and will not hog your system's resource. Code can be optimized for any Intel architecture processor from the 386 to the Pentium, including the new Pentium Pro chip. VisualAge includes the Win32s library, providing everything you need to target Windows 3.1 systems. Advanced C++ features such as run-time type information (RTTI) are available, and the C compiler supports structured exception handling. To produce the most efficient code possible, VisualAge for C++ uses advanced optimization techniques such as:

- Instruction scheduling
- Global register allocation
- User code inlining
- Intermodule optimizations

However, the benefits of efficient code can be reduced by inefficient use of resources such as memory. The new memory management algorithms used in the C and C++ run times are highly efficient in their use of memory, dramatically reducing the amount of memory overhead in your program.

If you need to minimize how much disk space your application uses, VisualAge for C++ offers an "optimize for size" option that minimizes the size of the code produced, but still runs efficiently. Disk space can also be reduced with our smart linking support that will find and remove unused code from your executable. While generating code for debugging, you can save disk space with our new debug packing support, which can drastically reduce the size of debug-ready executables without any loss of debug information. Also, the "line number only" option provides reduced debug support, but generates an executable only slightly larger than a program without any debug support.

Memory leaks are a common problem in C and C++ programming, and one of the most difficult types to diagnose. We've extensively enhanced our memory debugging support to help you find those persistent problems. For example, VisualAge for C++ run time provides full debug support of heap and leak detection.

Memory management has also been enhanced by providing support for multiple heaps and shared memory. VisualAge for C++ also includes support for user heaps such as transparent allocations from private heaps using malloc/new. This capability gives you even more flexibility in implementing your applications.

Finally, conformance to the following industry standards protects your investment and allows you to write portable code:

- ANSI C X3.159-1989 and ISO 9899:1990 (1992) C conformance
- Japanese MIA standards conformance
- C++ Draft Standard X3J16 Sept 1992 (plus RTTI)
- NIST Standard FIPS PUB 160C.

### Direct-to-SOM (DTS)

Direct-to-SOM (DTS) is an exciting new technology that combines familiar and powerful standard C++ syntax with the robustness and portability of IBM's System Object Model (SOM). DTS is not a tool, but a standard for implementing flexible and reusable objects, supported by the tools of VisualAge for C++.

To build a SOM object, it was once necessary to go through the time-consuming process of:

- writing IDL (Interface Definition Language)
- generating C++ bindings with the SOM compiler
- compiling C++ files with the C++ compiler

Now you can generate SOM objects at the current SOM 2.1 level that OS/2 uses, directly from the C++ compiler, simply by turning on a compiler option. And the compiler will also generate the corresponding OMG CORBA-compliant IDL for inter-language or DSOM applications. The VisualAge Browser shows SOM objects

in a different color than C++ objects. The data access builder has options that generate SOM classes (or IDL code). Best of all, the VisualAge Debugger now lets you browse and debug SOM objects as easily as regular C++ objects.

DTS has several advantages over native C++ programs:

- **Release to release binary compatibility (RRBC)** breaks the tight dependency between the code that implements a class and the client code that uses it. RRBC allows you to create and deploy a new version of a class with added function or data members, and even inherit from new base classes, without requiring an unchanged client to be recompiled. By packaging your SOM class in a DLL, you can replace an old DLL with the new one, and all applications that used it will continue to run.
- **Extensive dynamic facilities** allow you to query the properties of objects and classes, and the use of classes and methods whose names are not known until execution time. This allows a degree of flexibility and configurability familiar to programmers using Smalltalk or the OS/2 Workplace Shell. Applications can be extended by incrementally installing new classes that the application is told about through a configuration file or even by end user input.

If you are an experienced SOM user, who programmed SOM from C or non-DTS C++, you will welcome the power and flexibility that DTS provides. And, because you write C++ directly, you can use C++ features in your SOM classes used by C++ clients that weren't available before DTS; features like templates, operators, constructors with parameters, default parameters, static members, and more.

## Performance Execution Trace Analyzer - Time and Tune Your Code

This tool enables you to time and tune your applications, analyze program hangs and deadlocks, view multithread interactions, and better understand your code. It gives you a view of your program's run-time behavior you just can't get through debuggers or browsers. By collecting execution trace data and presenting it in several graphical diagrams, Performance Analyzer can provide information on:

- **Timing and tuning:** The trace file contains a detailed record of function calls and returns. Performance Analyzer can display the trace in a chronologically-scaled format. This helps you find the "hot spots" in the code *and* the cause of those hot spots.
- **Program hangs and deadlocks:** Performance Analyzer provides a complete history of the events leading up to the point the program stopped. You can view the function call stack any place in the application.
- **Multi-thread interactions:** You can look at the sequencing of procedures across threads, which can make problems with critical sections visible.

## To Help You Test and Debug...

### VisualAge Debugger - Find and Fix Coding Errors Fast

Your ability to develop robust software quickly and efficiently is directly related to how fast you can find and fix coding errors. The debugger is your primary tool for this process.

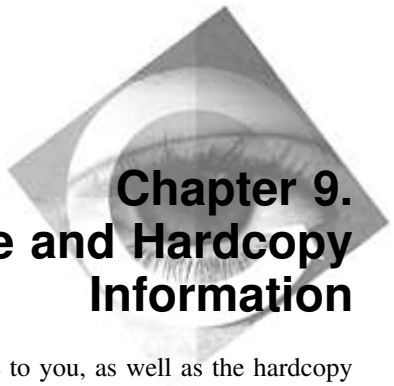
Efficiency comes from debugging at the source level, which means that you can look at your code exactly as you wrote it. Efficiency also comes from an optimized user interface that provides access to all the common debugger functions with a single mouse click; these include step, run, set/reset breakpoints, monitor variables, display call stack, display registers, and display storage.

In addition to all the functions commonly expected in a state-of-the-art debugger, the Debugger also provides an array of unique built-in tools to help locate problems and fix code quickly:

- Debug on demand enables you to open a debugging session whenever an error occurs in your application. For example, if an unhandled exception occurs and the debug-on-demand feature is enabled, the debugger is notified that an error has occurred. The debugger starts and attaches to your application at the point of fault. This can save you time because you do not have to recreate errors. Your application will run at full speed without any interference from the debugger until an exception is encountered.

  VisualAge C++ can start debug on demand for any application that fails while it is running, even if the application does not contain debug information. With debug on demand, it is possible to find and fix the problem in the application and let it continue running.
- C++ debugging features include template support, locate overloaded functions, class display, and debugging code in include files.
- Automatic heap checking helps to isolate memory management problems by checking for memory overwriting each time your program stops executing.
- Supports multithread Windows NT / Windows95 applications. It also can debug Win32s applications running on a Windows 3.11 system remotely.

Naturally, since it is fully integrated with the rest of the tools of VisualAge for C++, you can edit, browse, or recompile your code directly from your debugger session.



# Chapter 9. VisualAge for C++ Online and Hardcopy Information

This section describes the online information available to you, as well as the hardcopy books that come with IBM VisualAge for C++ for Windows.

## Online Information

You can access the online information in several ways:

- From the **Help** menu in any tool.
- From the Editor, Browser, Debugger, Performance Analyzer, Data Access Builder, or Visual Builder interface, by putting your cursor on a keyword and pressing F1 (for the Editor) or Ctrl-H (for the others).
- By issuing the **iview** command from a command line. The installation routine stores the online document files in the `\IBMCPPW\HELP` directory. To view the *Language Reference*, for example, make `C:\IBMCPPW\HELP` your current directory (substituting the drive where you installed VisualAge for C++ for `C:`) and enter the following command:

  ```
  IVIEW CPPLNG.INF
  ```

  If you want to get information on a specific topic, you can specify a word or a series of words after the filename. If the words appear in an entry in the table of contents or the index, the online document is opened to the associated section. For example, if you want to read the section on operator precedence in the *Language Reference*, you can enter the following command:

  ```
  IVIEW CPPLNG.INF OPERATOR PRECEDENCE
  ```

In the **Help** menu, in addition to the usual help topics, we have added a **How Do I** choice for the tool you are currently using, as well as entries for all of the online books. We have divided the online information into different categories to make it easier to find what you need. See Figure 7 on page 48.

© Copyright IBM Corp. 1993, 1996

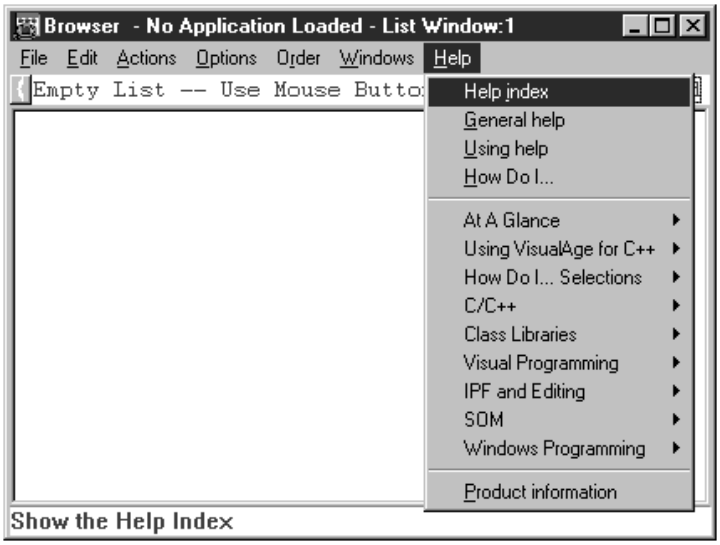



*Figure 7.*

Select the arrow next to the menu choice to see the books in that category. (They are described on the following pages.)

- **At A Glance** provides general information about VisualAge for C++.
- **Using VisualAge for C++** provides information about how to use the VisualAge for C++ tools.
- **How Do I... Selections** provides step-by-step instructions on how to perform various tasks.
- **C/C++** provides details on C and C++ programming techniques, as well as language constructs and library functions.
- **Class Libraries** describes the IBM Open Class Library.
- **Visual Programming** describes how to use Visual Builder to create applications.
- **IPF and Editing** provides information on using the Editor and creating online information.
- **SOM** describes how to create SOM applications.
- **Windows Programming** provides online help for windows programming.

## At a Glance

*Installation Guide and Product Overview*

Gives you details on how to install VisualAge for C++ and a general overview of the product.

*Frequently Asked Questions*

Provides answers to the questions that our customers ask most often.

### Using VisualAge for C++

*User's Guide*
Tells you how to use the VisualAge for C++ tools, including the WorkFrame, editor, compiler, linker, debugger, browser, Performance Analyzer, and utilities. It also describes the options and commands for these tools.

Each choice in the **Using VisualAge for C++** menu takes you to the appropriate section in the *User's Guide*. For example, selecting **Browsing** opens the *User's Guide* to the section on using the Browser.

### How Do I... Selections

Each choice in the menu takes you to step-by-step instructions for performing tasks with that tool. For example, selecting **Debugger** takes you to the task help for the debugger. Selecting **Overall** opens the help for general VisualAge for C++ tasks that involve more than one tool.

To see step-by-step instructions for the tool you are currently using, select **How do I**... from the first grouping of topics in the **Help** menu.

The desktop icons for the How Do I help are in the **How Do I** folder.

### C/C++

*Programming Guide*
Describes different C and C++ programming techniques, such as building DLLs and using templates.

*C Library Reference*
Describes the C library functions.

*Language Reference*
Describes the language elements of both C and C++.

### Class Libraries

*Open Class Library User's Guide*
Tells you how to use the class libraries that comprise IBM Open Class in your C++ programs.

*Open Class Library Reference*
Describes the class libraries that comprise IBM Open Class.

## Visual Programming

*Visual Builder User's Guide*
: Describes how to use Visual Builder to create your applications graphically, without writing code.

*Visual Builder Parts Reference*
: Describes the reusable parts for Visual Builder (parts are like classes).

*Building Parts for Fun and Profit*
: Tells you how to create your own libraries of parts for others to use.

## IPF and Editing

*IPF User's Guide*
: Describes how to edit, compile, link, debug, analyze, browse, and internationalize your applications

*IPF Programmer's Guide and Reference*
: Tells you how to create online information using the IPF help facility.

*Editor Command Reference*
: Describes how to customize and program the VisualAge for C++ editor.

## SOM

*SOM Programming Guide*
: Describes the System Object Model (SOM) and how to use it.

*SOM Programming Reference*
: Provides details about the SOM interfaces.

## Windows Programming

Win32
: Provides detailed information on utilizing Win32 APIs and services specific to the Windows NT and Windows 95 operating systems within your application.

If you prefer books to online information you can order **most** of this information in hardcopy format through the same channels you ordered the VisualAge for C++ product.

## Hardcopy Information

All VisualAge for C++ packages contain this book (*Installation Guide and Product Overview*) and:

*License Information*
: Details the license agreement.

*HELP!! and how to get it*
: Tells you what to do if you have a problem with VisualAge for C++.

If you have the CD-ROM-only version of VisualAge for C++, these are the only hardcopy books you have. All other information is online in .INF and PostScript format.

If you have the CD-ROM document package version of VisualAge for C++, you get the following additional hardcopy information:

*Programming Guide*
: Describes C and C++ programming techniques for both novice and experienced users.

*User's Guide*
: Tells you how to use the WorkFrame, editor, compiler, linker, debugger, browser, Performance Analyzer, and utilities.

*Visual Builder User's Guide*
: Describes how to use Visual Builder to create your applications graphically, without writing code.

*Open Class Library User's Guide*
: Tells you how to use the IBM Open Class libraries in your C++ programs, including the Collection, Database Access, and User Interface classes.

If you want to order even more books you can do this through the same channels you ordered the VisualAge for C++ product.

The hardcopy information comes in the following two packages:

Library Group 1, 33H4981

- *User's Guide*
- *Programming Guide*
- *Open Class Library User's Guide*
- *Visual Builder User's Guide*

Library Group 2, 33H4982

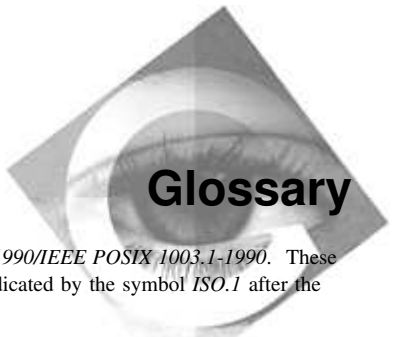
- *Language Reference*
- *C Library Reference*
- *Open Class Library Reference: Volume 1*
- *Open Class Library Reference: Volume 2*
- *Open Class Library Reference: Volume 3*
- *Open Class Library Reference: Volume 4*
- *Visual Builder Parts Reference*
- *Building VisualAge for C++ Parts for Fun and Profit*

If you have VisualAge for C++ on CD-ROM, you can also print the PostScript files for the books, as described in "PostScript Files."

## PostScript Files

If you have VisualAge for C++ on CD-ROM, PostScript files are available for all information that can be ordered in hardcopy. The files are located in the PS directory. You can print these files on any PostScript printer.

To print the files drag the file icon and drop it on your printer icon.



# Glossary

This glossary defines terms and abbreviations that are used in this book. Included are terms and definitions from the following sources:

- *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Such definitions are indicated by the symbol *ANSI* after the definition.
- *IBM Dictionary of Computing*, SC20-1699. These definitions are indicated by the registered trademark *IBM* after the definition.
- *X/Open CAE Specification. Commands and Utilities, Issue 4. July, 1992*. These definitions are indicated by the symbol *X/Open* after the definition.
- *ISO/IEC 9945-1:1990/IEEE POSIX 1003.1-1990*. These definitions are indicated by the symbol *ISO.1* after the definition.
- *The Information Technology Vocabulary*, developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol *ISO-JTC1* after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol *ISO Draft* after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

---

## A

**access**. An attribute that determines whether or not a class member is accessible in an expression or declaration. It can be public, protected, or private.

**action**. A description of tool or function that can be used to manipulate a project's parts, or build a project's target. Examples are compile, link, and edit.

**application**. (1) The use to which an information processing system is put; for example, a payroll application, an airline reservation application, a network application.*IBM*. (2) A collection of software components used to perform specific types of user-oriented work on a computer.*IBM*.

**array**. An aggregate that consists of data objects, with identical attributes, each of which may be uniquely referenced by subscripting.

**ASCII (American National Standard Code for Information Interchange)**. The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

**Note:** IBM has defined an extension to ASCII code (characters 128-255).

**AVL tree**. A balanced binary search tree that does not allow the height of two siblings to differ by more than one.

## B

**B*-tree (B star tree)**. A tree in which only the leaves contain whole elements. All other nodes contain keys.

**balance**. (1) For audio, refers to the relative strength of the left and right channels. A balance level of 0 is left channel only. A balance level of 100 is right channel only (2) A state of equilibrium, usually between treble and bass.

**based on**. A relationship between two classes in which one class is implemented through the other. A new class is "based on" an existing class when the existing class is used to implement it.

**block**. (1) In programming languages, a compound statement that coincides with the scope of at least one of the declarations contained within it. A block may also specify storage allocation or segment programs for other purposes. *ISO-JTC1*. (2) A string of data elements recorded or transmitted as a unit. The elements may be characters, words, or physical records. *ISO Draft*. (3) The unit of data

© Copyright IBM Corp. 1993, 1996

transmitted to and from a device. Each block contains one record, part of a record, or several records.

**build**. An action that invokes the WorkFrame Build tool. The Build tool manages the project's makefile, as well as build dependencies between projects in a project hierarchy.

**built-in**. A function that the compiler automatically puts inline instead of generating a call to the function. Synonymous with predefined. *IBM*.

# C

**C++ class library**. See class library.

**C++ library**. A system library that contains common C++ language subroutines for file access, memory allocation, and other functions.

**call**. To transfer control to a procedure, program, routine, or subroutine. *IBM*.

**character**. (1) A letter, digit, or other symbol that is used as part of the organization, control, or representation of data. A character is often in the form of a spatial arrangement of adjacent or connected strokes. *ANSI*. (2) A sequence of one or more bytes representing a single graphic symbol or control code. This term corresponds to the ISO C standard term *multibyte character* (multi-byte character), where a single-byte character is a special case of the multi-byte character. Unlike the usage in the ISO C standard, *character* here has no necessary relationship with storage space, and *byte* is used when storage space is discussed. *X/Open. ISO.1*.

**class**. (1) A group of objects that share a common definition and that therefore share common properties, operations, and behavior. (2) A C++ aggregate that may contain functions, types, and user-defined operators in addition to data. Classes can be defined hierarchically, allowing one class to be an expansion of another, and classes can restrict access to their members.

**C library**. A system library that contains common C language subroutines for file access, string operators, character operations, memory allocation, and other functions. *IBM*.

**class library**. A collection of classes.

**collection**. (1) In a general sense, an implementation of an abstract data type for storing elements. (2) An abstract class without any ordering, element properties, or key properties. All abstract Collection Classes are derived from Collection.

**COM**. Component Object Model.

**command**. A request to perform an operation or run a program. When parameters, arguments, flags, or other operands are associated with a command, the resulting character string is a single command.

**Compound Document Framework**. A starting point for creating a server or container document component that is OLE-enabled. Compound documents allow for the integration of arbitrary or unstructured data from different sources into one location.

**Compound Object Model (COM)**. The underlying model for all OLE services. It consists of a variety of APIs and object interfaces that allow container components to communicate and interact with one another.

**condition**. (1) A relational expression that can be evaluated to a value of either true or false. *IBM*. (2) An exception that has been enabled, or recognized, by the *Language Environment* and thus is eligible to activate user and language condition handlers. Any alteration to the normal programmed flow of an application. Conditions can be detected by the hardware/operating system and result in an interrupt. They can also be detected by language-specific generated code or language library code.

**container**. An object that holds other objects.*IBM.*

**control**. A graphic object that represents operations or properties of other objects. See also tree control.

**cursor**. A reference to an element at a specific position in a data structure.

# D

**data type**. The properties and internal representation that characterize data.

**definition**. (1) A data description that reserves storage and may provide an initial value. (2) A declaration that allocates storage, and may initialize a data object or specify the body of a function.

**degree**. The number of children of a node.

**delete**. (1) A C++ keyword that identifies a free-storage deallocation operator. (2) A C++ operator used to destroy objects created by operator new.

**device**. A computer peripheral or an object that appears to the application as such. *X/Open. ISO.1*.

**direct manipulation**. (1) A user interface technique whereby the user initiates application functions by manipulating the objects, represented by icons, on the Presentation Manager (PM) or Workplace Shell desktop. The user typically initiates an action by:

1. Selecting an icon
2. Pressing and holding down a mouse button while "dragging" the icon over another object's icon on the desktop
3. Releasing the mouse button to "drop" the icon over the target object.

Thus, this technique is also known as "drag and drop" manipulation. (2) In Windows, a unit of data. The unit is often part of a task that is shared among users.

**directory**. A type of file containing the names and controlling information for other files or other directories. *IBM*.

**display**. To direct the output to the user's terminal. If the output is not directed to the terminal, the results are undefined. *X/Open*.

**dynamic**. Pertaining to an operation that occurs at the time it is needed rather than at a predetermined or fixed time. *IBM*.

**dynamic link library (DLL)**. A file containing executable code and data bound to a program at load time or run time. The code and data in a dynamic link library can be shared by several applications simultaneously.

# E

**element**. The component of an array, subrange, enumeration, or set.

**element equality**. A relation that determines whether two elements are equal.

**environment variable**. Any of a number of variables that describe the way an operating system is going to run and the devices it is going to recognize.

**exception**. (1) A user or system error detected by the system and passed to an operating system or user exception handler. (2) For C++, any user, logic, or system error detected by a function that does not itself deal with the error but passes the error on to a handling routine (also called "throwing the exception").

**exception handling**. A type of error handling that allows control and information to be passed to an exception handler when an exception occurs. In C++, try, catch, and throw expressions are the constructs used to implement C++ exception handling.

**extension**. (1) An element or function not included in the standard language. (2) File name extension.

# F

**filter**. (1) A command whose operation consists of reading data from standard input or a list of input files and writing data to standard output. Typically, its function is to perform some transformation on the data stream. (2) In WorkFrame, the value of a type. The filter of a type can be expressed as a file mask; a regular expression; a logical-OR, logical-AND, or logical-NOT of a list of types; or a filter determined by a PAM.

**folder**. A directory.

**frame**. A border around a window.

**function**. A named group of statements that can be called and evaluated and can return a value to the calling statement. *IBM*.

**function call**. An expression that moves the path of execution from the current function to a specified function and evaluates to the return value provided by the called function. A function call contains the name of the function to which control moves and a parenthesized list of values. *IBM*.

# G

**global**. Pertaining to information available to more than one program or subroutine. *IBM*.

**graphical user interface (GUI)**. Type of computer interface consisting of a visual metaphor of a real-world scene, often of a desktop.

**graphics**. A picture defined in terms of graphic primitives and graphic attributes.

**GUI**. Graphical user interface.

# H

**hash table**. A data structure that divides all elements into (preferably) equal-sized categories, or buckets, to allow quick access to the elements. The hash function determines which bucket an element belongs in.

**heap**. An unordered flat collection that allows duplicate elements.

# I

**inheritance**. (1) An object-oriented programming technique that allows you to use existing classes as bases for creating other classes. (2) A mechanism by which a derived class can use the attributes, relationships, and member functions defined in more abstract classes related to it (its base classes). See also multiple inheritance.

**instance (of a class)**. An object that is a member of that class. An object created according to the definition of that class.

**instruction**. A program statement that specifies an operation to be performed by the computer, along with the values or locations of operands. This statement represents the programmer's request to the processor to perform a specific operation.

**instruction scheduling**. An optimization technique that reorders instructions in code to minimize execution time.

**I/O Stream Library**. A class library that provides the facilities to deal with many varieties of input and output.

# K

**kernel**. The core of an operating system, usually responsible for basic I/O and process execution.

**keyword**. (1) A predefined word reserved for the C or C++ language that you cannot use as an identifier. (2) A symbol that identifies a parameter.

# L

**label**. An identifier within or attached to a set of data elements.
*ISO Draft*.

**leaves**. In a tree, nodes without children. Synonymous with terminals.

**library**. (1) A collection of functions, function calls, subroutines, or other data. (2) A set of object modules that can be specified in a link command.

**link**. To interconnect items of data or portions of one or more computer programs; for example, linking of object programs by a linkage editor to produce an executable file.

**linker**. A program that resolves cross-references between separately compiled object modules and then assigns final addresses to create a single executable program.

**local**. (1) In programming languages, pertaining to the relationship between a language object and a block such that the language object has a scope contained in that block. *ISO-JTC1*. (2) Pertaining to that which is defined and used only in one subdivision of a computer program. *ANSI*.

# M

**macro**. An identifier followed by arguments (may be a parenthesized list of arguments) that the preprocessor replaces with the replacement code located in a preprocessor `#define` directive.

**make**. An action in which a project's target is built from a makefile by a make utility.

**message**. A request from one object that the receiving object implement a method. Because data is encapsulated and not directly accessible, a message is the only way to send data from one object to another. Each message specifies the name of the receiving object, the method to be implemented, and any parameters the method needs for implementation.

**method**. Synonym for member function.

**MIDI**. Musical Instrument Digital Interface. A standard used in the music industry for interfacing digital musical instruments.

**migrate**. To move to a changed operating environment, usually to a new release or version of a system. *IBM*.

**mix**. (1) An attribute that determines how the foreground of a graphic primitive is combined with the existing color of graphics output. Also known as foreground mix. Contrast with background mix. (2) The combination of audio or video sources during postproduction.

**mixer**. A device used to simultaneously combine and blend several inputs into one or two outputs.

**mode**. A collection of attributes that specifies a file's type and its access permissions. *X/Open. ISO.1.*

**module**. A program unit that usually performs a particular function or related functions, and that is distinct and identifiable with respect to compiling, combining with other units, and loading.

**Monitor**. A window that displays output from monitored actions. The Monitor window is attached to the project container.

**multibyte character**. A mixture of single-byte characters from a single-byte character set and double-byte characters from a double-byte character set.

**multimedia**. Computer-controlled presentations combining any of the following: text, graphics, animation, full-motion images, still video images, and sound.

**multiple inheritance**. (1) An object-oriented programming technique implemented in C++ through derivation, in which the derived class inherits members from more than one base class. (2) The structuring of inheritance relationships among classes so a derived class can use the attributes, relationships, and functions used by more than one base class.

See also inheritance.

**multithread**. Pertaining to concurrent operation of more than one path of execution within a computer.

# N

**name**. In the C++ language, a name is commonly referred to as an identifier. However, syntactically, a name can be an identifier, operator function name, conversion function name, destructor name, or qualified name.

**native**. The rendering mechanism and format (RMF) that best represents the object and is the best one for rendering. For example, a native of Cincinnati understands the streets in the area better than someone who has just moved there. Therefore, a Cincinnati native can get from point A to point B quicker than a newcomer. Likewise, a native RMF can get the data transferred from point A to point B more efficiently than the additional RMFs. We can use additional RMFs when we cannot use the native, or optimal, approach.

# O

**object**. (1) A computer representation of something that a user can work with to perform a task. An object can appear as text or an icon. (2) A collection of data and member functions that operate on that data, which together represent a logical entity in the system. In object-oriented programming, objects are grouped into classes that share common data definitions and member functions. Each object in the class is said to be an instance of the class. (3) In Visual Builder, an instance of an object class consisting of attributes, a data structure, and operational member functions. It can represent a person, place, thing, event, or concept. Each instance has the same properties, attributes, and member functions as other instances of the object class, though it has unique values assigned to its attributes. (4) In Windows, any item that is or can be linked into another Windows application, such as a sound, graphic, piece of text, or portion of a spreadsheet. An object must be from an application that supports OLE. See object linking and embedding (OLE).

**object linking and embedding (OLE)**. (1) An API that supports compound documents, cross-application macro control, and common object registration. OLE defines protocols for in-place editing, drag-and-drop data transfers, structured storage, custom controls, and more. (2) A data-sharing scheme that allows dissimilar applications to create single complex documents cooperatively. The documents can consist of material that a single application could not have created on its own.

**object-oriented programming**. A programming approach based on the concepts of data abstraction and inheritance. Unlike procedural programming techniques, object-oriented programming concentrates on what data objects comprise the problem and how they are manipulated, not on how something is accomplished.

**OLE**. See object linking and embedding.

**operator precedence**. In programming languages, an order relation defining the sequence of the application of operators within an expression. *ISO-JTC1.*

# P

**parameter**. (1) In the C and C++ languages, an object declared as part of a function declaration or definition that acquires a value on entry to the function, or an identifier following the macro name in a function-like macro definition. *X/Open*. (2) Data passed between programs or procedures. *IBM*.

**part**. In Visual Builder, a part is a self-contained object with a standardized public interface consisting of a set of external features that allow the part to interact with other parts. A part is implemented as a class that supports the INotifier protocol and has a part interface defined.

**path name**. (1) A string that is used to identify a file. A path name consists of, at most, {PATH_MAX} bytes, including the terminating null character. It has an optional beginning slash, followed by zero or more file names separated by slashes. If the path name refers to a directory, it may also have one or more trailing slashes. Multiple successive slashes are considered to be the same as one slash. A path name that begins with two successive slashes may be interpreted in an implementation-dependent manner, although more than two leading slashes will be treated as a single slash. The interpretation of the path name is described in *pathname resolution*. *ISO.1*. (2) A file name specifying all directories leading to the file.

**pointer**. A variable that holds the address of a data object or function.

**portability**. The ability of a programming language to compile successfully on different operating systems without requiring changes to the source code.

**precedence**. The priority system for grouping different types of operators with their operands.

**process**. (1) A collection of code, data, and other system resources, including at least one thread of execution, that performs a data processing task. (2) A running application, its address space, and its resources. (3) An instance of a running program. A Win32 process owns a 4-GB address space containing the code and data for an application's .exe file; it does not execute anything. It also owns certain resources, such as files, dynamic memory allocations, and threads. (4) A program running under OS/2, along with the resources associated with it (memory, threads, file system resources, and so on).

**program**. (1) One or more files containing a set of instructions conforming to a particular programming language syntax. (2) A self-contained, executable module. Multiple copies of the same program can be run in different processes.

**project**. (1) A container that groups related objects (tasks) into a primary window. When the user opens the object, the object has its own primary window. (2) In WorkFrame, the complete set of data objects (called project parts) and actions needed to build a single target, such as a dynamic link library (DLL) or executable file (EXE).

**prototype**. A function declaration or definition that includes both the return type of the function and the types of its arguments.

# R

**return**. A language construct that ends an execution sequence in a procedure.

**root**. A node that has no parent. All other nodes of a tree are descendants of the root.

**RTTI**. Run-time type identification.

**run-time type identification (RTTI)**. A mechanism in the C++ language for determining the class of an object at run time. It consists of two operators, one for determining the run-time type of an object (**`typeid`**) and one for doing type conversions that are checked at run time (**`dynamic_cast`**). A **`type_info`** class describes the RTTI available and defines the type returned by the **`typeid`** operator.

# S

**sequence**. A sequentially ordered flat collection.

**shell**. A program that interprets sequences of text input as commands. It may operate on an input stream or it may interactively prompt and read commands from a terminal. *X/Open*.

This feature is provided as part of OpenEdition MVS Shell and Utilities feature licensed program.

**silent mode**. See unattended mode.

**source file**. A file that contains source statements for such items as high-level language programs and data description specifications. *IBM*.

**stack**. A data structure in which new elements are added to and removed from the top of the structure. A stack is characterized by Last-In-First-Out (LIFO) behavior.

**statement**. An instruction that ends with the character **;** (semicolon) or several instructions that are surrounded by the characters { and }.

**static**. A keyword used for defining the scope and linkage of variables and functions. For internal variables, the variable has block scope and retains its value between function calls. For external values, the variable has file scope and retains its value within the source file. For class variables, the variable is shared by all objects of the class and retains its value within the entire program.

**stream**. (1) A continuous stream of data elements being transmitted, or intended for transmission, in character or binary-digit form, using a defined format. (2) A file access object that allows access to an ordered sequence of characters, as described by the ISO C standard. A stream provides the additional services of user-selectable buffering and formatted input and output.

**string**. A contiguous sequence of characters.

**structured exception handling**. A Windows-specific mechanism for handling system exceptions that matches exceptions with handlers based on the value returned from an exception filter expression.

**subsystem**. A secondary or subordinate system, usually capable of operating independently of or asynchronously with, a controlling system. *ISO Draft*.

# T

**task**. (1) In a multiprogramming or multiprocessing environment, one or more sequences of instructions treated by a control program as an element of work to be accomplished by a computer. *ISO-JTC1. ANSI*. (2) A routine that is used to simulate the operation of programs. Tasks are said to be *nonpreemptive* because only a single task is executing at any one time. Tasks are said to be *lightweight* because less time and space are required to create a task than a true operating system process.

**template**. A family of classes or functions where the code remains invariant but operates with variable types.

**thread**. (1) The smallest unit or path of execution within a process. *IBM*. (2) A piece of executing code. (3) In Windows, each thread is allocated its own stack from the owning process' 4-GB address space, and each one has its own set of processor registers, called the thread's context. See also primary thread and zero page thread.

**tool bar**. The area under the title bar that displays the tools available.

**tree**. A hierarchical collection of nodes that can have an arbitrary number of references to other nodes. A unique path connects every two nodes.

**type**. (1) The description of the data and the operations that can be performed on or by the data. See also *data type*. (2) In WorkFrame, describes a group of project files of parts in terms of an expression, such as file masks, regular expressions, or a list of other types, logical-OR'd.

# U

**unattended mode**. A mode in which no operator is present or in which no operator station is included at system generation.

# V

**variable**. In programming languages, a language object that may take different values, one at a time. The values of a variable are usually restricted to a certain data type. *ISO-JTC1*.

**VGA**. Video graphics adapter.

**video**. Pertaining to the portion of recorded information that can be seen.

**video graphics adapter (VGA)**. A graphics controller for color displays. The pel resolution of the video graphics adapter is 4:4.

**visible**. Visibility of identifiers is based on scoping rules and is independent of *access.*

# W

**Win32**. The name of a 32-bit application programming interface (API).

See also Win32 API.

**Win32 API**. (1) A set of Win32 functions that can be called from source code. (2) A 32 bit-bit version of the 16-bit Windows 3.1 API (native to Windows NT).

See also Win32.

**Win32s**. A platform that the Win32 API is implemented on. (The s stands for subset.) It consists of a virtual-device

driver and dynamic link libraries (DLLs) that add the Win32 API to the 16-bit Windows 3.n system. It includes structured exception handling and limited implementations of memory-mapped files.

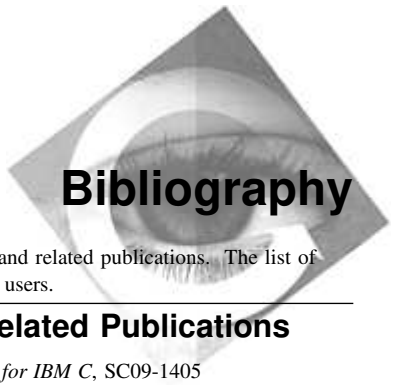
See also Win32 API and Windows 95.

**Windows NT**. A platform that the Win32 API is implemented on. It is a portable, high-end operating system, which can run several different types of applications simultaneously. It is the only Win32 platform for machine architectures based on processors other than the x86, and it supports multiple processors.

See also Win32 API.

**Windows 95**. (1) A 32-bit operating system that allows you to run 32-bit application. Windows 95 is a multitasking, multithreaded operating system that can control multiple programs at once. Each program can have multiple concurrent threads or independently executing subcomponents. (2) A platform that the Win32 API is implemented on. It supports image color matching, modems, and other services. It partially supports asynchronous file I/O, debugging, registry, security, and event-logging functions.

**write**. (1) To output characters to a file, such as standard output or standard error. Unless otherwise stated, standard output is the default output destination for all uses of the term *write*. *X/Open*. (2) To make a permanent or transient recording of data in a storage device or on a data medium. *ISO-JTC1*. *ANSI*.



# Bibliography

This bibliography lists the publications that make up the IBM VisualAge for C++ library and related publications. The list of related publications is not exhaustive but should be adequate for most VisualAge for C++ users.

## The IBM VisualAge for C++ Library

The following books are part of the IBM VisualAge for C++ library.

- *Installation Guide and Product Overview*, S33H-5030
- *User's Guide*, S33H-5031
- *Programming Guide*, S33H-5032
- *Visual Builder User's Guide*, S33H-5034
- *Visual Builder Parts Reference*, S33H-5035
- *Building VisualAge for C++ Parts for Fun and Profit*, S33H-5036
- *Open Class Library User's Guide*, S33H-5033
- *Open Class Library Reference*, S33H-5039
- *Language Reference*, S33H-5037
- *C Library Reference*, S33H-5038
- *SOM Programming Guide*, S33H-5043
- *SOM Programming Reference*, S33H-5044

## C and C++ Related Publications

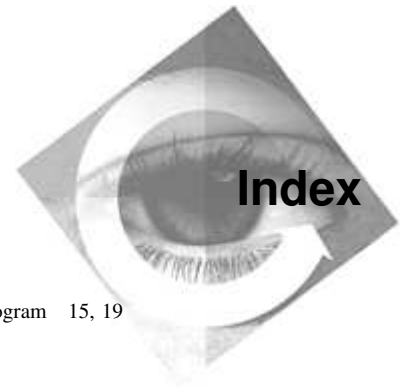
- *Portability Guide for IBM C*, SC09-1405
- *American National Standard for Information Systems / International Standards Organization — Programming Language C (ANSI/ISO 9899-1990[1992])*

### Non-IBM Publications

Many books have been written about the C++ language and related programming topics. The authors use varying approaches and emphasis. The following is a sample of some non-IBM C++ publications that are generally available. This sample is not an exhaustive list. IBM does not specifically recommend any of these books, and other C++ books may be available in your locality.

- *The Annotated C++ Reference Manual* by Margaret A. Ellis and Bjarne Stroustrup, Addison-Wesley Publishing Company.
- *C++ Primer* by Stanley B. Lippman, Addison-Wesley Publishing Company.
- *Object-Oriented Design with Applications* by Grady Booch, Benjamin/Cummings.
- *Object-Oriented Programming Using SOM and DSOM* by Christina Lau, Van Nostrand Reinhold.

© Copyright IBM Corp. 1993, 1996



# Index


## A
about this booklet 2
adding components 17
administrator tasks
  preparing the server 5
autoexec.bat 4

## B
books in VisualAge for C++ library 51

## C
CD-ROM installation
  overview 7
  performance characteristics 7
choosing an installation method 7
choosing an installation target 9
components
  adding 17
  adding after initial installation 17
  choosing what to install 17
  deleting 22
  descriptions 25
Custom installation
  overview 8
  performance characteristics 8

## D
deleting
  components 22
  VisualAge for C++ 22
description of VisualAge for C++ 1, 35
desktop folder
  rebuilding 21
directory
  choosing an installation target 9
disk space
  determining available space 13
  determining required space 12, 18
  requirements 3
display requirements 3

## E
exiting the install program 15, 19

## F
features 1, 35

## G
getting help for installation 7
getting started after installation 25

## H
hardware requirements 3
help for installation 7

## I
information and books
  hardcopy publications 51
  online 47
  PostScript printable files 52
installation methods
  CD-ROM installation 7
  choosing a method 8
  Custom installation 8
  Minimal installation 7
  Typical installation 7
installing
  additional components 17
  choosing a method 8
  choosing a target directory 9
  condensed instructions 9
  detailed instructions 10
  methods 7
  overview 7
  problem determination 29
  Win32s support 15

## L
LAN installation
  administrator tasks 5


© Copyright IBM Corp. 1993, 1996

LAN installation *(continued)*
description 5
preparing the server 5

## M

memory requirements 3
Minimal installation
overview 7
performance characteristics 7

## O

overview of installation 7

## P

performance characteristics
CD-ROM installation 7
Custom installation 8
Minimal installation 7
Typical installation 7
portability
publications 61
PostScript files 52
preinstallation tasks 3
printing VisualAge for C++ PostScript files 52
product description 1, 35
publications
related 61

## Q

quitting the install program 15, 19

## R

RAM, minimum 3
related publications
portability 61
VisualAge for C++ 61
requirements, hardware and software 3

## S

selecting components to install 17
server, setting up as installation base 5
setting target directory 12, 13
stopping the installation 15, 19

## T

target drive and directory
setting 12, 13
time requirements 4
tour of VisualAge for C++ 27
tutorial 27
Typical installation
overview 7
performance characteristics 7

## U

updating desktop folders and icons 21

## V

VisualAge for C++
publications 61

## W

Win32s support
installing 15

# Communicating Your Comments to IBM

IBM VisualAge for C++ for Windows
Installation Guide and Product Overview
Version 3.5

Publication No. S33H-5030-00

If there is something you like—or dislike—about this book, please let us know. You can use one of the methods listed below to send your comments to IBM. If you want a reply, include your name, address, and telephone number. If you are communicating electronically, include the book title, publication number, page number, or topic you are commenting on.

The comments you send should only pertain to the information in this book and its presentation. To request additional publications or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give it to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:
  - United States and Canada: 416-448-6161
  - Other countries: (+1)-416-448-6161
- If you prefer to send comments electronically, use the network ID listed below. Be sure to include your entire network address if you wish a reply.
  - Internet: torrcf@vnet.ibm.com
  - IBMLink: toribm(torrcf)
  - IBM/PROFS: torolab4(torrcf)
  - IBMMAIL: ibmmail(caibmwt9)

# Readers' Comments — We'd Like to Hear from You

**IBM VisualAge for C++ for Windows**
**Installation Guide and Product Overview**
**Version 3.5**

**Publication No. S33H-5030-00**

**Overall, how satisfied are you with the information in this book?**

| | Very Satisfied | Satisfied | Neutral | Dissatisfied | Very Dissatisfied |
|---|---|---|---|---|---|
| Overall satisfaction | □ | □ | □ | □ | □ |

**How satisfied are you that the information in this book is:**

| | Very Satisfied | Satisfied | Neutral | Dissatisfied | Very Dissatisfied |
|---|---|---|---|---|---|
| Accurate | □ | □ | □ | □ | □ |
| Complete | □ | □ | □ | □ | □ |
| Easy to find | □ | □ | □ | □ | □ |
| Easy to understand | □ | □ | □ | □ | □ |
| Well organized | □ | □ | □ | □ | □ |
| Applicable to your tasks | □ | □ | □ | □ | □ |

**Please tell us how we can improve this book:**

Thank you for your responses. May we contact you? □ Yes □ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.

**Readers' Comments — We'd Like to Hear from You**
S33H-5030-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Canada Ltd. Laboratory
Information Development
2G/345/1150/TOR
1150 EGLINTON AVENUE EAST
NORTH YORK ONTARIO CANADA M3C 1H7

Fold and Tape **Please do not staple** Fold and Tape



Cut or Fold
Along Line

Part Number: 33H5030
Program Number: 33H4979
33H4980

Printed in U.S.A.