C++ Builder 4
Client/Server Foundations

Courseware Manual
Chapter 1

Version 1.0

Chapter 1: A Tour of C++ Builder

What will be covered in this chapter:

- ❑ The definition of C++ Builder
- ❑ The C++ Builder environment
- ❑ The C++ Builder menu system
- ❑ C++ Builder tools, including the Object Inspector and the Form Designer
- ❑ Code Insight
- ❑ How to configure your C++ Builder environment
- ❑ Installing your own tools within C++ Builder

C++ Builder Overview

C++ Builder is one of the most powerful ways to create general purpose Windows applications. While it is being touted as the best way to create Client/Server applications, it is also a terrific environment to create more general purpose, non-database applications. You could create a spreadsheet package or word processor in C++ Builder. There are no intrinsic limits to its capabilities. You could even create an application development environment in C++ Builder, but Inprise has already beaten you to it.

There is a long list of what makes C++ Builder such an appealing product. It embodies all of the things that programmers have learned in finding best ways to approach Windows application development. Here is a partial list of the unique feature set of C++ Builder:

- The Professional AppBrowser IDE visual development environment to manage the creation of applications
- The excellent Borland C++ Compiler
- Stand-alone executables, with no run-time libraries required to distribute applications
- Produce reusable DLLs
- Fully object-oriented, with a well respected and rock solid language base
- The ability to create native components within C++ Builder
- Visual, two-way tools
- Exception handling support in the language, which allows the creation of robust, production quality applications
- A fully integrated editor and debugger
- Inherent database connectivity through the Borland Database Engine
- An integrated industrial strength report writer with live data in design mode
- The Object Repository for storing form, datamodule, and project templates
- Painless upsizing from local databases to client/server databases
- The ability to create Multi-Threaded Windows 95/NT applications
- Code Insight to help coding and reduce syntax errors

There are three distinct flavors of C++ Builder available: the Standard, the Professional and the Enterprise editions.

Standard Edition

The Standard edition of C++ Builder includes most of the features that we will discuss in this course. Actually, the base product is exactly the same no matter which edition you utilize. The difference is in the tools and additional software that you receive in each package. The standard edition includes the following features:

- C++ Builder IDE and compiler (including a command line compiler)
- Borland Database Engine, which includes support for dBASE and Paradox table formats
- Dockable Toolbars to create Toolbar and Windows Docking in applications
- Visual Component Library (VCL) with more than 85 customizable components
- Database Desktop

Professional Edition

Inprise designed the Professional edition of C++ Builder for programmers who need more functionality than the Standard edition provides, but do not need all the extra database features of the Enterprise package. It comes with everything from the standard edition plus:

- Source code for the VCL
- InstallShield express
- InterBase server
- ODBC Connectivity
- Database Explorer
- Open Tools API
- Hooks for team development with InterSolv's PVCS
- Sample OCX's for creating charts, spellchecking and more
- The ability to create ActiveX controls and ActiveForms
- Internet Solutions Pack to web-enable applications

Enterprise Edition

The Enterprise edition of C++ Builder includes all of the above features with the following additions and enhancements:

- SQL Links native drivers for Oracle, Sybase, InterBase, Informix, DB2, and MS SQL Server
- SQL Database Explorer for managing server data
- SQL Monitor
- Visual Query Builder
- Data Migration Tool
- InterBase database server with 4 user license
- InterSolv PVCS for team development
- DecisionCube components to create multi-dimensional CrossTabs
- Multi-tier Distributed Application Services Suite to develop multi-tier applications

- One-Step CORBA objects
- Full support for Microsoft Transaction Server (MTS)

The actual C++ Builder development environment is exactly the same base IDE no matter which edition you have. The only difference is in the features that are provided. In fact, you may buy the Professional edition, then later upsize to the Enterprise edition.

The C++ Builder Environment

The C++ Builder environment is what you see when you start C++ Builder. You will notice immediately that C++ Builder is not a MDI (Multiple Document Interface) application like most Windows 3.x programs. A MDI application consists of a parent window frame that includes one or many smaller windows inside the parent window. A perfect example of a MDI application is the Windows Program Manager in Windows 3.x. In MDI applications all of the child windows are framed or bounded by the parent window. C++ Builder, on the other hand, is a modified SDI (Single Document Interface) application, which means that there are no nested windows. Generally, Windows95 programs are going to use the SDI interface instead of the MDI model found in Windows 3.x. See Chapter 25 for more details.

Main Menu bar

C++ Builder's main menu bar is shown below:

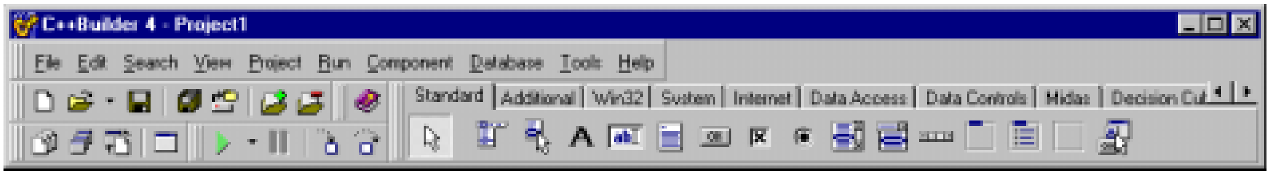

C++Builder 4 - Project1
File Edit Search View Project Run Component Database Tools Help
Standard Additional Win32 System Internet Data Access Data Controls Midas Decision Cu

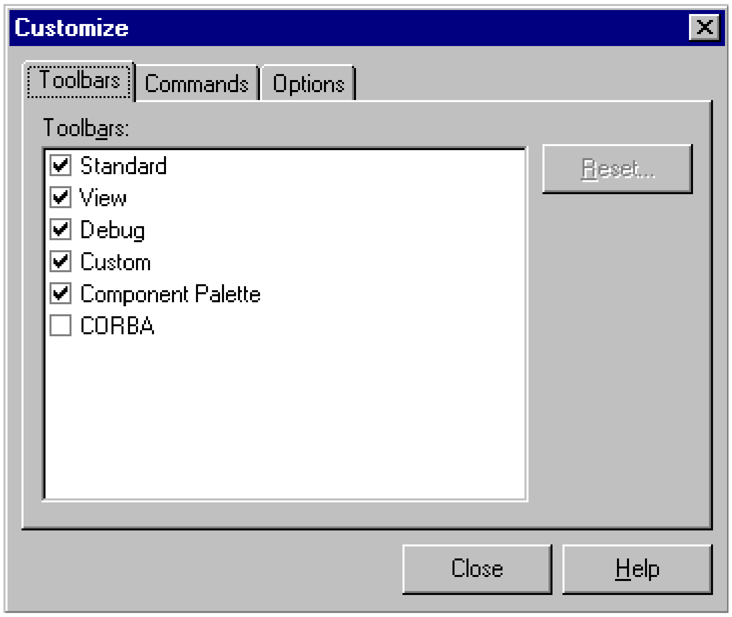
Figure 1.1 - Main Menu bar

The menu system is standard Windows, with pull down menus, cascading menus, and dialog boxes. We will not go into detail about the menus here—each of the menu options will be discussed in separate sections.

Toolbars and Speed Buttons

C++ Builder has several toolbars on the Main Menu bar. One toolbar is used for the main menu system described above. Another toolbar displays the Component Palette, which is discussed in the next section. There are also four more toolbars that are used to display speed buttons, which are shortcuts for various menu selections. These speed buttons adhere to Microsoft's style for the speed button toolbars in their Office 97 products. The buttons look flat until the mouse cursor is dragged over them. When this occurs, the speed buttons become 'raised'. The speed buttons in C++ Builder also have tool tips, which are the little floating yellow boxes that appear if you let the mouse linger over one of the buttons long enough. This interface trick is becoming standard in Windows programs, and it helps considerably in interpreting what a speed bar button's picture face means.

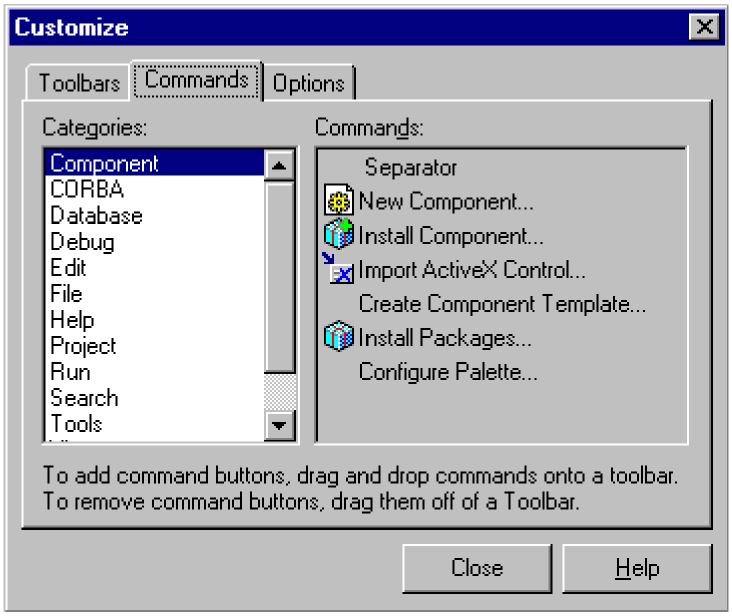
Like almost every aspect of C++ Builder, the toolbars are abundantly configurable. To reposition the toolbars, simply grab the bar located on the left side of the toolbar and drag it to its new position. You can also configure which speed buttons are displayed on each toolbar by right-clicking on any of the toolbars and choosing Customize… from the speed menu. Your selection leads to the following dialog box:



☑ Standard
☑ View
☑ Debug
☑ Custom
☑ Component Palette
☐ CORBA

Figure 1.2 – Customize Dialog – Toolbars Page

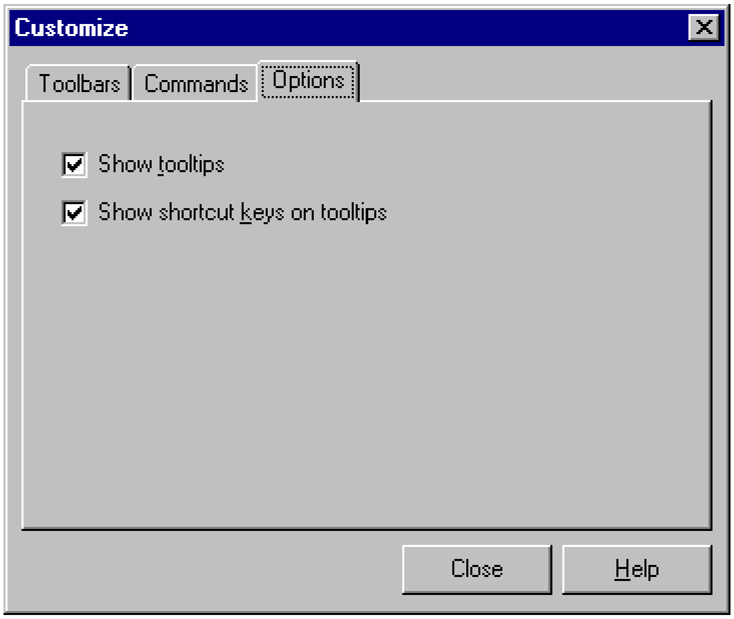
The Toolbars page in this dialog allows you to turn a toolbar ‘on’ or ‘off’ by selecting the checkbox next to the corresponding toolbar. Also, if at any time you wish to revert changes made to a toolbar back to their default state, the Reset button can be clicked for that toolbar.



To add command buttons, drag and drop commands onto a toolbar.
To remove command buttons, drag them off of a Toolbar.

Figure 1.3 - Customize Dialog – Commands Page

The Commands page in the dialog allows you to change or add commands to toolbars. If you want to add a command item to the toolbar, grab it with the left mouse button down and drag it up to the toolbar. It is now anchored on the toolbar. Notice that there are categories on the left that correspond to the menus in which these commands reside. So, almost any command that is available in a menu is also available to be dropped on a toolbar.



☑ Show tooltips

☑ Show shortcut keys on tooltips

Figure 1.4 - Customize Dialog – Options Page

The Options page in this dialog allows you to configure the tooltips associated with the speed buttons. You can choose whether or not to display the tooltips as well as to display the shortcut keys for the command.

The Component Palette

The remainder of the main application form is taken up by the component palette:

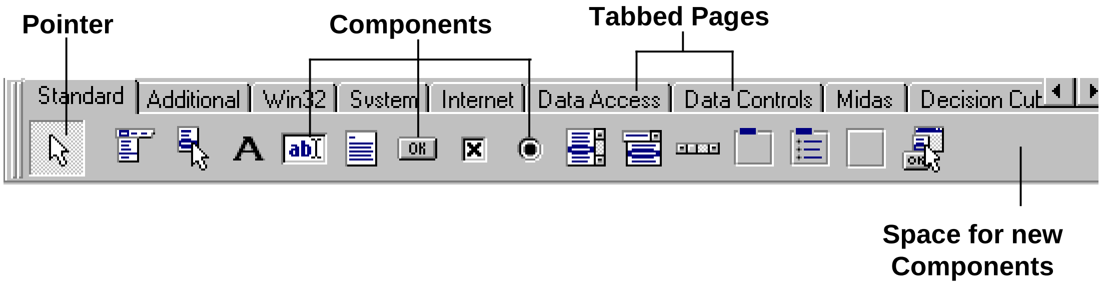


Figure 1.5 - The Component Palette

The component palette shows all of the controls, both visible and invisible, that can be placed on a form. Each tab contains components grouped by functionality. For example, the components shown in the above picture are all standard Windows components (the use of which will be discussed later). Notice that there is space for additional components. One of the revolutionary things about C++ Builder is that you can create your own components using C++. The place holders are places that you can populate with your own components as your experience with C++ Builder grows.

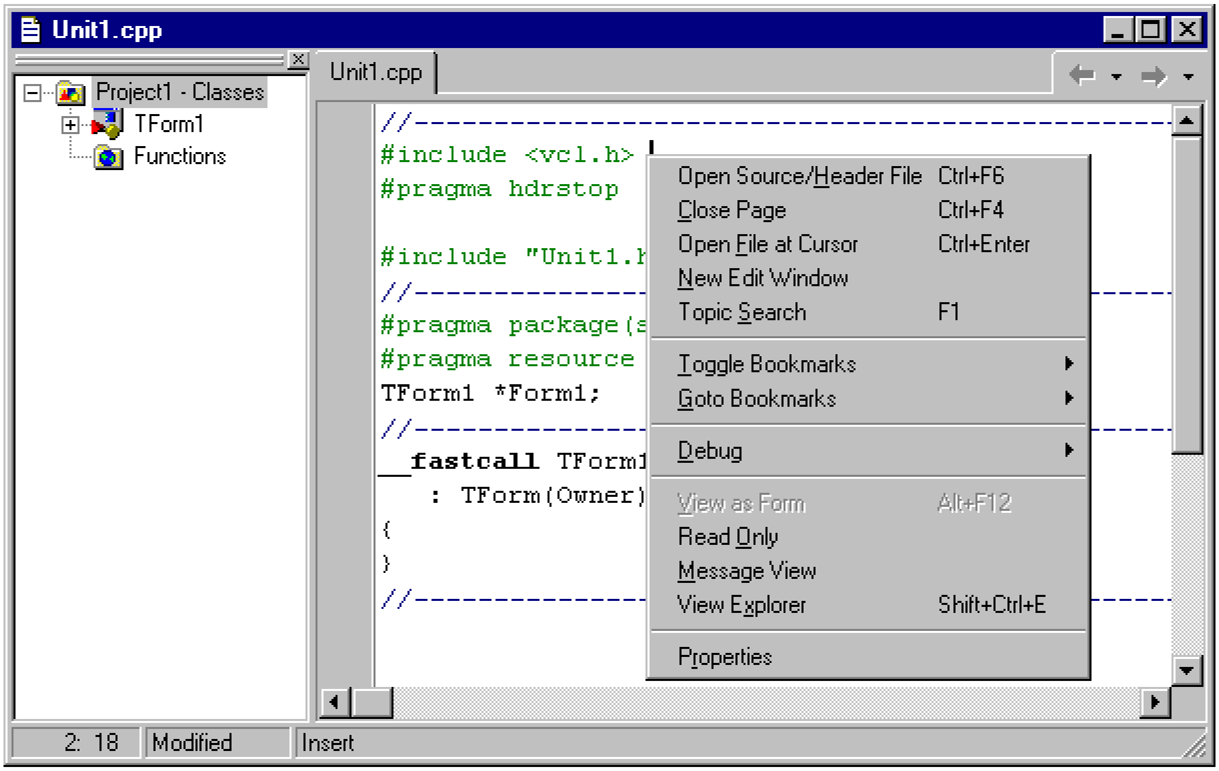
You may have also noticed that the right-hand side of the component palette has a right arrow if you fill the available space on a given tab. You can use the arrow to scroll and see the hidden components. If you scroll to the right to see any hidden components, a left arrow will be displayed on the left-hand side of the component palette for scrolling back.

Not only can you create new components and place them on a tab, you can create new tabs on which to place components. Also, if you create more tabs than will fit on the screen, you can use the left and right arrows located on the top right of the component palette to scroll to see the additional tabs.

Notice that one of the component palette tabs is for ActiveX controls. ActiveX controls are fairly generic components that are designed to work in the 32-bit world. C++ Builder uses all ActiveX controls as if they were built-in components. So, you can buy existing ActiveX controls instead of writing them yourself, or use ActiveX controls that you may already own without modification. However, these controls are, by their nature, not as flexible as the components that were written in C++ Builder. ActiveX controls require the associated ActiveX file to be distributed with the application.

Speed Menus

One of the ubiquitous interface elements that you see in C++ Builder is speed menus, which is another interface element that is becoming popular in Windows products. A speed menu is a menu that pops up next to your cursor when you click once on the right mouse button, and they are generally very specific to the task at hand. This is the speed menu that pops up in the code editor:



```
//-----------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h
//---------------                                              ------
#pragma package(s
#pragma resource
TForm1 *Form1;
//---------------                                              ------
__fastcall TForm1
  : TForm(Owner)
{
}
//---------------                                              ------
```

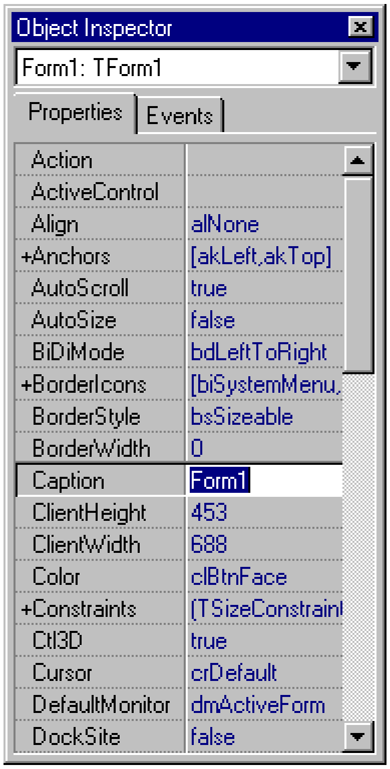
Figure 1.6 - Speed Menu in code editor

This menu has a lot of specific options for the editor. Each speed menu is different, depending on where it was invoked. For example, changing the default speed bar buttons is accomplished by right-clicking on a toolbar. This pops up a speed menu whose options allow you to turn ‘on’ or ‘off’ the Standard, View, Debug, Custom, Corba, or Component Palette toolbars, or choose the Customize command.

Speed menus are extremely handy. You should become accustomed to right-clicking when you need to do something in the IDE because there is a good chance there is a speed menu item available for doing it.

Object Inspector

The Object Inspector is a window that appears to the left of the form design window by default and is the primary interface element for setting properties and assigning event code.



Form1: TForm1

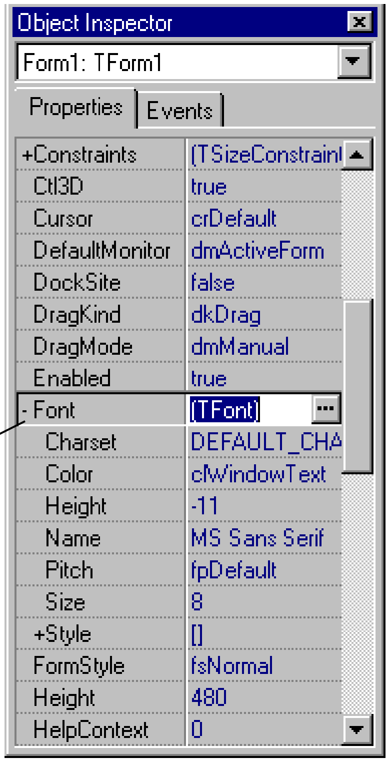
Action	
ActiveControl	
Align	alNone
+Anchors	[akLeft,akTop]
AutoScroll	true
AutoSize	false
BiDiMode	bdLeftToRight
+BorderIcons	[biSystemMenu,
BorderStyle	bsSizeable
BorderWidth	0
Caption	Form1
ClientHeight	453
ClientWidth	688
Color	clBtnFace
+Constraints	(TSizeConstrain
Ctl3D	true
Cursor	crDefault
DefaultMonitor	dmActiveForm
DockSite	false

Figure 1.7 - Object Inspector

Properties

Properties refer to aspects of a given control that can be changed either at design time or at run time. Essentially, properties are attributes that determine how a component looks, acts, and reacts. You will use the Object Inspector to change properties during design time. We will talk at great length about setting properties, and what the properties are used for later in the course.

The combo box at the top of the Object Inspector indicates what control's properties are being inspected. You use the Object Inspector to view and change the properties of all items on forms, and view and define event handlers for form controls. The "properties" tab page has an alphabetical list of the properties associated with the control. Likewise, the page accessed through the "events" tab lists the events associated with the control. Some of the properties have a "+" sign next to them. You can think of those items as outline items, in which the "+" indicates there are invisible sub-entries, and the "-" signifies you can see all the sub-entries. For example, here is the Object Inspector with the *Font* property expanded:



+Constraints	(TSizeConstrainl
Ctl3D	true
Cursor	crDefault
DefaultMonitor	dmActiveForm
DockSite	false
DragKind	dkDrag
DragMode	dmManual
Enabled	true
- Font	(TFont)
Charset	DEFAULT_CHA
Color	clWindowText
Height	-11
Name	MS Sans Serif
Pitch	fpDefault
Size	8
+Style	[]
FormStyle	fsNormal
Height	480
HelpContext	0

Expanded Font Property

Figure 1.8 - Expanded Font Property

This sort of hierarchical approach to properties makes it easier for the user to set properties that are related to one another. Because all of these properties are aspects of the font, they are all included under the font outline item. Notice that this arrangement can go to as many levels deep as necessary. For example, the *Style* property under *Font* is also expandable.

Another thing to notice in the above screen shot is the ellipsis button on the far right of the entry field for the *Font* property. The ellipsis ("...") is used to invoke the property editor for this property. The property editor is different for each property. For example, if you click on the ellipsis to choose a font, the standard Windows font picker dialog box appears:

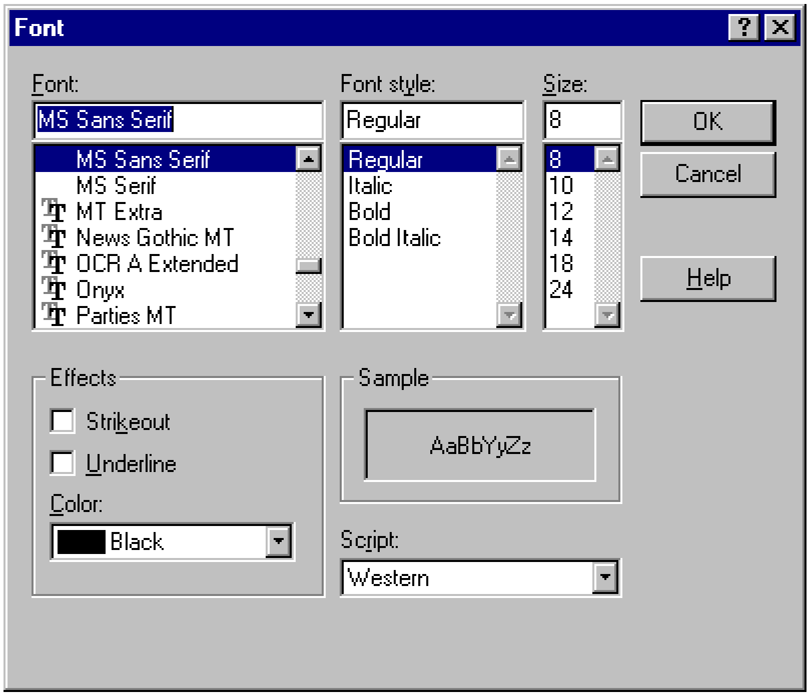


Figure 1.9 - Windows Font dialog box

Another very nice feature supported by the Object Inspector is the ability to double-click on a field that has a limited number of options to scroll through the possible options. For example, for properties that are either true or false, you can double-click to toggle the value back and forth. If the property is a color, and you have a list of five possible colors, double-clicking in the Object Inspector will toggle through all of the possible values.

Events

“Events” is the other tab on the Object Inspector’s panel. The events' page for the above control is shown in Figure 1.10:

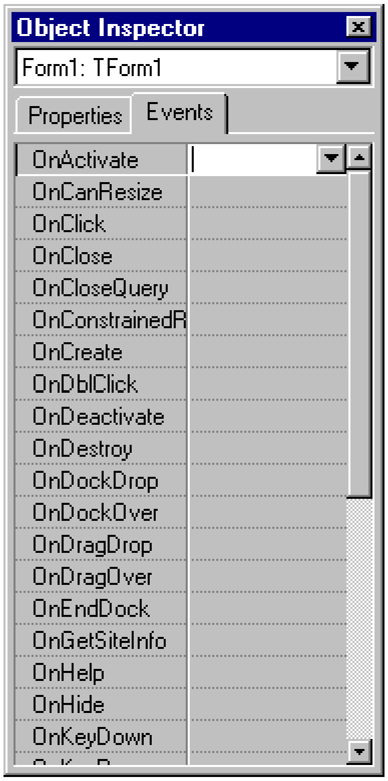


Figure 1.10 - The Events Page

We will give a more formal definition of events, later. For now, events are groups of code you provide that determine what action is taken at run-time for the given Windows message or event. You can invoke the program editor by double-clicking within a particular event field; or, if there is already a routine that you want to run in response to the event, you can click on the down arrow and pick it from a list.

Events and the handling of events are one of the broadest topics in C++ Builder. We will return to the events page later in the course.

Form Designer, Code Editor and Class Explorer

The Form Designer has several chapters devoted to it, so we will not get into great depth about it here. However, it is usually one of the first things that you notice about C++ Builder.

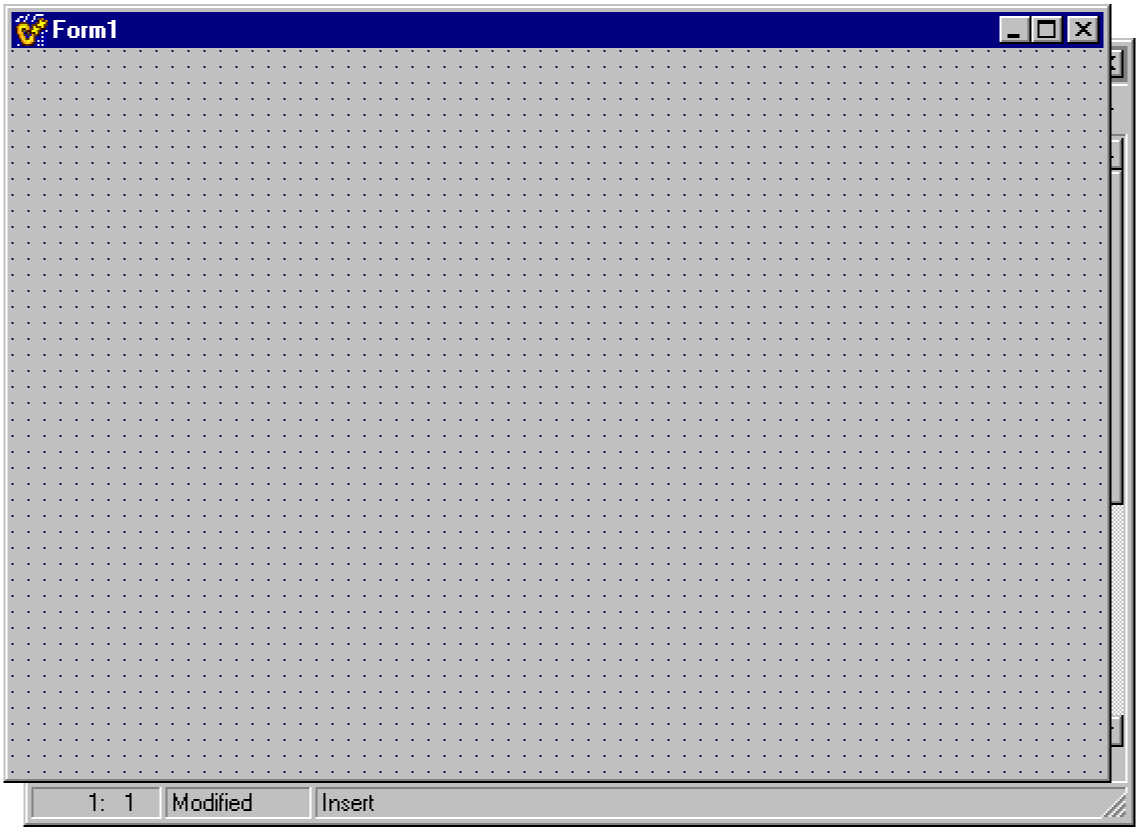
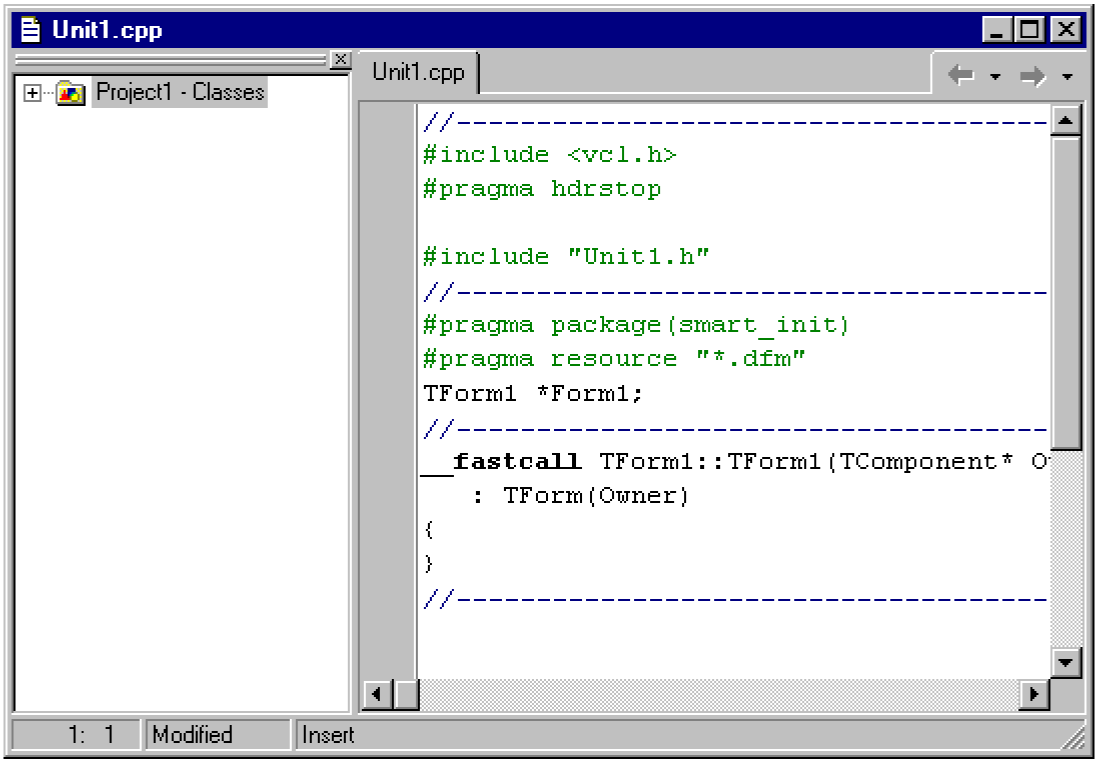


Figure 1.11 - The Form Designer

Because C++ Builder is a SDI application, whatever other applications that are open can be seen “underneath” these floating palettes. Also, as all of these windows are autonomous, you can minimize each of them onto the desktop, where they appear just like minimized applications. However, when you minimize the entire C++ Builder desktop (via the minimize button on the main menu bar), it “sucks up” all open panels and all of the C++ Builder tools appear as a single minimized application.

Code Editor and Class Explorer

The Code Editor and Class Explorer windows allow the programmer to create, edit and navigate through the code that makes up their application.
The tabs at the top switches you from unit to unit if you have more than one program file open. F12 is the key that makes it very easy to switch back and forth between the Form Designer and Code Editor. There is also a speed bar button to exchange the form for its source file, which is handy if you are designing a form that completely obscures the editor tab.



```
//------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//------------------------------------------
__fastcall TForm1::TForm1(TComponent* O
    : TForm(Owner)
{
}
//------------------------------------------
```

Figure 1.12 – Class Explorer and Code Editor

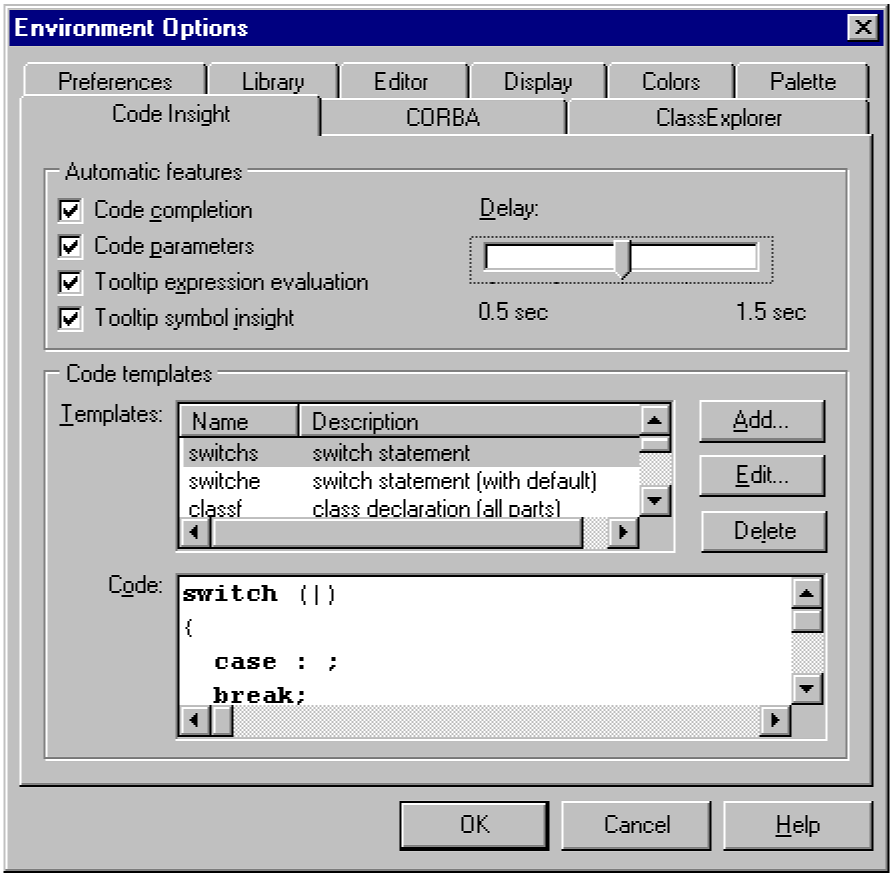
The Code Editor is a multi-page window that can include the Class Explorer window docked inside it (as shown above). Or, the Class Explorer can be undocked as a floating window. Like other C++ Builder tools, the Code Editor is fully customizable through the **Tools | Environment Options** dialog box (discussed below). The Code Editor offers several features for the programmer to utilized:

- Brief-style editing
- Color syntax highlighting
- Several editing commands
- Code Insight

The Class Explorer is a new addition to C++ Builder that allows you to navigate through your unit files. It has a tree view that shows all the types, classes, properties, methods, global variables, and global routines defined in a unit. Like most standard tree views, you can expand or collapse the nodes on the tree. The unit that is currently open in the Code Editor has its information displayed in the Class Explorer. Both the Code Editor and the Class Explorer will be discussed in greater detail over the duration of this course.

Code Insight

The C++ Builder environment helps programmers produce code more rapidly and with fewer errors by providing them with Code Insight wizards. The Code Insight features can be configured in the **Tools | Environment Options** dialog on the Code Insight page.



☑ Code completion
☑ Code parameters
☑ Tooltip expression evaluation
☑ Tooltip symbol insight

Name	Description
switchs	switch statement
switche	switch statement (with default)
classf	class declaration (all parts)

```
switch (|)
{
  case : ;
  break;
```

Figure 1.13 – Code Insight Page

Code Insight offers programmers four features:

- Code Completion
- Code Parameters
- Code Templates
- ToolTip Expression Evaluation

Code Completion

Code Completion will display for the programmer a list of properties, methods and events for a class or an object defined by a class in code. To invoke this list, the programmer should type in the class name or object name followed by a period. The list will be displayed after the specified delay setting in the Environment Options dialog. This list can also be invoked by using **Ctrl+Spacebar.**

You can also display a list of valid arguments for a procedure, function, or method call using **Ctrl+Spacebar,** as well as getting a list of valid arguments when an assignment statement is used for variable.

Code Templates

You can create new or edit existing Code Templates in the Environment Options dialog. Code Templates allow the programmer to insert common statements into their code quickly by simply pressing **Ctrl+J.** A list of the currently defined code templates will be displayed for you to choose from.

Code Parameters

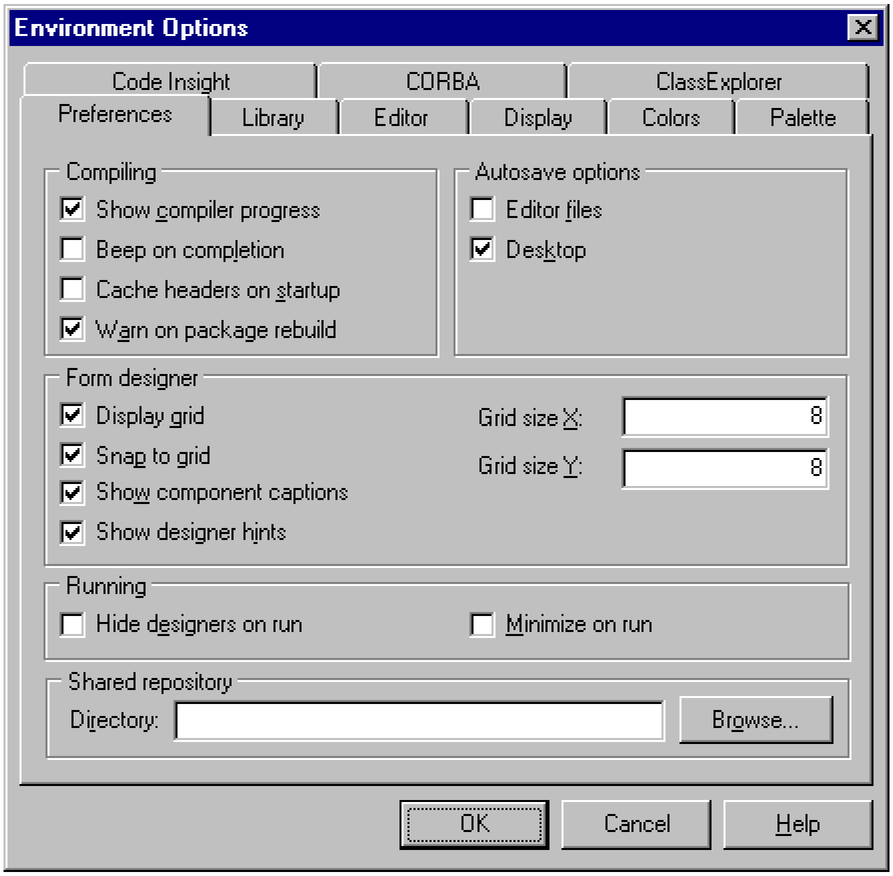
Code Parameters displays a list of required arguments for a method you enter in code followed by an open parenthesis.

ToolTip Expression Evaluation

While debugging your source code, you can see the value of a variable in a tooltip box by enabling Tooltip Expression Evaluation. Simply place the mouse pointer over a variable while stepping through your code to see the tooltip.

Environment Configuration

C++ Builder's desktop, editor, and all of its tools are very configurable. Most of these configuration options are on one large tabbed dialog box, invoked by selecting **Tools | Environment Options...** :



Compiling
☑ Show compiler progress
☐ Beep on completion
☐ Cache headers on startup
☑ Warn on package rebuild

Autosave options
☐ Editor files
☑ Desktop

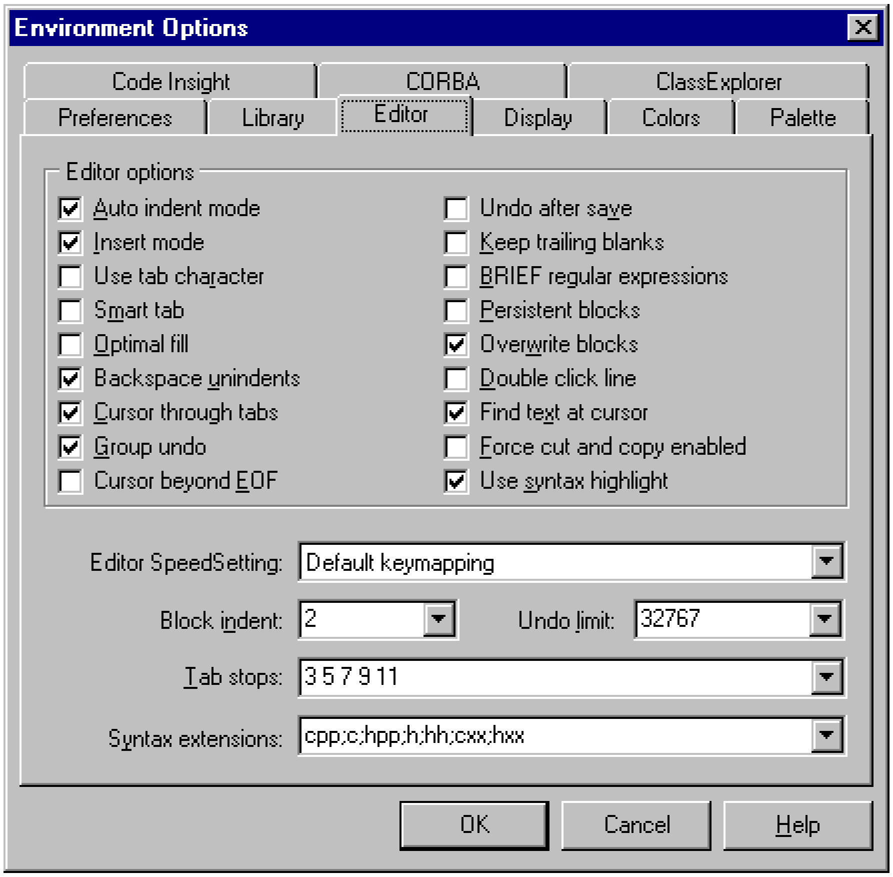
Form designer
☑ Display grid
☑ Snap to grid
☑ Show component captions
☑ Show designer hints
Grid size X: 8
Grid size Y: 8

Running
☐ Hide designers on run
☐ Minimize on run

Shared repository
Directory:

Figure 1.14 - Environment Options dialog box

This dialog allows you to set whichever of C++ Builder's options you want to change. In particular, the editor is customizable to your heart's content. Since Inprise owns the programming editor Brief, they have included many of its advanced features.



Editor options

☑ Auto indent mode
☑ Insert mode
☐ Use tab character
☐ Smart tab
☐ Optimal fill
☑ Backspace unindents
☑ Cursor through tabs
☑ Group undo
☐ Cursor beyond EOF
☐ Undo after save
☐ Keep trailing blanks
☐ BRIEF regular expressions
☐ Persistent blocks
☑ Overwrite blocks
☐ Double click line
☑ Find text at cursor
☐ Force cut and copy enabled
☑ Use syntax highlight

Editor SpeedSetting: Default keymapping

Block indent: 2

Undo limit: 32767

Tab stops: 3 5 7 9 11

Syntax extensions: cpp;c;hpp;h;hh;cxx;hxx

Figure 1.15 - Brief settings on editor page

Most of these options are self-explanatory, and there is always excellent on-line help for every aspect of the environment. However, the Tab Stops section of the above dialog may look strange to those of you who are not Brief users. The long sequence of numbers is the actual column positions of tab stops. However, in the Brief column numbering scheme, the first column is 1 (rather than 0), so the example above puts one tab stop starting at column 3 (which is actually just two spaces from the left margin).

Installing Tools

C++ Builder is an open environment, meaning that Inprise has created a special API (application programming interface) for programmers to integrate their products right into C++ Builder. This way you cannot tell that a separate product is running. A good example of this is PVCS Version Control software. C++ Builder has special hooks built-in for PVCS. In fact, when you install C++ Builder, the install program asks you if you have PVCS, and if you want it incorporated into C++ Builder. The reason C++ Builder has hooks already for PVCS is because the C++ Builder development team used PVCS to control C++ Builder versioning.

Other tool vendors are taking advantage of this Open Tools API, including the MKS version control package. Expect to see many more tools touting the claim that they integrate completely within C++ Builder.

Even without special hooks for a tool, it is easy to install your favorite tools right into the C++ Builder environment. The **Tools** menu from the C++ Builder main menu allows you to install stand-alone programs that can be launched from within C++ Builder by choosing them from this menu after they are installed. The default items control C++ Builder's configuration. To add your own tools to this menu, choose **Tools | Configure Tools...**, on the main C++ Builder menu. You will see the following dialog:

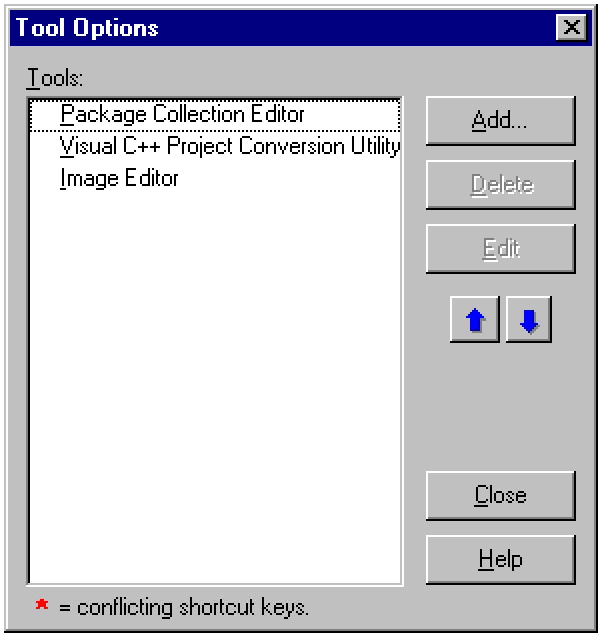

Tool Options
Tools:
Package Collection Editor
Visual C++ Project Conversion Utility
Image Editor
Add...
Delete
Edit
Close
Help
* = conflicting shortcut keys.

Figure 1.16 - Tools Options dialog box

This dialog gives you the ability to add any tools you like to the environment. Just choose Add and enter the appropriate information. C++ Builder will install the tool onto its **Tools** menu. The arrow keys on this dialog let you change the order that the items appear on the Tools menu. When you add programs, you can pass information from C++ Builder, such as a file name or extension, to the new program. The Add dialog includes a list of macro commands that can pass information to the program that you are installing.

Summary

What was covered in this chapter:

- ✔ We looked briefly at the C++ Builder menu structure, including its abundant use of speed menus.
- ✔ We examined toolbars, and how to configure them to suit your own taste. Almost any option that is available from a menu can be placed on the toolbars for quick access.
- ✔ We investigated the component palette, and talked briefly about how the component palette can be configured endlessly by adding your own components to it.
- ✔ We looked at the main C++ Builder tools—the Object Inspector, Form Designer, Code Editor, Class Explorer and Code Insight.
- ✔ We saw how easy it is to change the way some of the tools work through the Environment Options dialog.
- ✔ We discovered how to add your own favorite tool right into the C++ Builder menu system.

As you can see from this brief overview, C++ Builder is an easy-to-use tool. Starting with the next chapter, we will see that this ease of use sits atop a tool of unprecedented power.

CBUILDER 4 FOUNDATION COURSEWARE NO-NONSENSE LICENSE STATEMENT AND DISCLAIMER

IMPORTANT – READ CAREFULLY
This license statement and disclaimer constitutes a legal agreement (“License Agreement”) between you (an individual) and Inprise Corporation (“Inprise”) for the Cbuilder 4 Foundation Course ("Courseware"), including any software, media, and accompanying on-line or printed documentation. This License Agreement and your rights are only for each of the first twelve chapters of the Courseware that are being provided at no charge under this special program.

BY INSTALLING, COPYING, OR OTHERWISE USING THE COURSEWARE, YOU AGREE TO BE BOUND BY ALL OF THE TERMS AND CONDITIONS OF THE LICENSE AGREEMENT.

Upon your acceptance of the terms and conditions of the License Agreement, Inprise grants you the right to use the Courseware in the manner provided below.

This Courseware is owned by Inprise or its suppliers and is protected by copyright law and international copyright treaty. Therefore, you must treat this Courseware like any other copyrighted material (e.g., a book), except that you may either make one copy of the Courseware solely for backup or archival purposes or transfer the Courseware to a single hard disk provided you keep the original solely for backup or archival purposes.

You may transfer the Courseware and documentation on a permanent basis provided you retain no copies and the recipient agrees to the terms of the License Agreement. Except as provided in the License Agreement, you may not transfer, rent, lease, lend, copy, modify, translate, sublicense, time-share or electronically transmit or receive the Courseware, media or documentation.

WARRANTY DISCLAIMER
THE COURSEWARE ARE PROVIDED "AS IS," WITHOUT WARRANTY OF ANY KIND.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, BORLAND AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NON-INFRINGEMENT, WITH REGARD TO THE COURSEWARE. SOME STATES/JURISDICTIONS DO NOT ALLOW LIMITATIONS ON DURATION OF AN IMPLIED WARRANTY, SO THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

LIMITATION OF LIABILITY
TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL BORLAND OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE COURSEWARE PRODUCT OR THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF BORLAND HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, BORLAND'S ENTIRE LIABILITY UNDER ANY PROVISION OF THIS LICENSE AGREEMENT SHALL BE LIMITED TO THE GREATER OF THE AMOUNT ACTUALLY PAID BY YOU FOR THE COURSEWARE PRODUCT OR U.S. $25. BECAUSE SOME STATES AND JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

U.S. GOVERNMENT RESTRICTED RIGHTS

The Courseware and documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraphs (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (c)(1) and (2) of the Commercial Computer Software-Restricted Rights at 48 CFR 52.227-19, as applicable. Manufacturer is Inprise Corporation., 100 Enterprise Way, Scotts Valley, CA 95066.

GENERAL PROVISIONS
This statement may only be modified in writing signed by you and an authorized officer of Inprise. If any provision of this statement is found void or unenforceable, the remainder will remain valid and enforceable according to its terms. If any remedy provided is determined to have failed for its essential purpose, all limitations of liability and exclusions shall remain in effect.

This statement shall be construed, interpreted and governed by the laws of the State of California, U.S.A. This statement gives you specific legal rights; you may have others which vary from state to state and from country to country. Inprise reserves all rights not specifically granted in this statement.

Form 05/13/00