Introduction to Borland C++ Builder

Borland C++ Builder is a programming studio used to create computer applications for the Microsoft Windows operating systems. Borland C++ Builder is based on the C++ computer language with a lot of improvements and customized items.

Because this book is not about creating applications for Microsoft Windows, the version used is not important. This book addresses only console applications. When writing this book, I used Borland C++ Builder 6 Professional. Although I used version 6, the version is not important for this book. There are only two minor differences with the version versions, as far as console applications are concerned. The New item on the File menu now leads to Other... to access the New Items dialog box. There is also an additional radio button in the Console Wizard dialog box. This will not be used in this book.

Launching C++ Builder

There are various ways you can launch the program. The most common consists of clicking.

To create a shortcut on the desktop, in Microsoft Windows higher than Win95, click Start -> Programs -> Borland C++ Builder 6, right-click C++ Builder 6 and click Send To -> Desktop (Create Shortcut)

- To start Borland C++ Builder, click Start, position the mouse on Programs, position the mouse on Borland C++ Builder, C++ Builder 5
  Borland C++ Builder IDE

The Integrated Development Environment

An Integrated Development Environment (IDE) is an application that provides a friendly interface for creating computer programs.

Bcb's IDE is essentially a classic application. On top, there is a title bar that displays the name of the application and the program currently running. The title bar itself is made of three sections.

In this book, Bcb stands for Borland C++ Builder or C++ Builder

The Title Bar

1. Click the application icon . The system icon is used to identify the application that you are using. Almost every application has its own system icon. The system icon holds its own list of actions; for example, it can be used to move, minimize, maximize or close (when double-clicked) a window.
2. To see an example, while the system menu is displaying, click Minimize.
3. To bring back the IDE, on the Task bar, click C++ Builder

The main section of the title bar displays the C++ Builder name of the application, and the name of the program that is running. A C++ Builder program is called a project. When Bcb starts, it creates a starting project immediately, and it names the starting project, Project1. If or when you add other projects, they take subsequent names such as Project2, Project3, etc. This main section of the title bar is also used to move, minimize, maximize the top section of the IDE, or to close Bcb. On the right section of the title bar, there are three system buttons with the following roles

Button	Role
_ _	Minimizes the window
□ □	Maximizes the window
❐ ❐	Restores the window
✕ ✕	Closes the window

The Main Menu

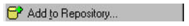
Under the title bar, there is a range of words located on a gray bar; this is called the menu or main menu.

In this book, the word "Main Menu" refers to the menu on top of the IDE.

To use a menu, you click one of the words and the menu expands.

1. Click File. There are four main types of menus you will encounter.

   When clicked, the behavior of a menu that stands alone depends on the actions prior to clicking it. Under the File menu, examples include Save, Close All or Exit. For example, if you click Close All, Bcb will find out whether the project had been saved already. If it has been, the project would be closed; otherwise, you would be asked whether you want to save it before closing it.
2. To see an example, click Save.
3. When you are asked to save, click Cancel.
4.

   A menu that is disabled is not accessible at the moment. This kind of menu depends on another action or the availability of something else.
   To see an example, click Edit and notice Undelete, Redo, Cut, Copy, and Paste.
5. 

   A menu with three dots means that an action is required in order to apply its setting(s). Usually, this menu would call a dialog box where the user would have to make a decision.

   As an example, on the main menu, click Tools and click Editor Options...
6. On the Editor Options dialog, click OK.
7.

   A menu with an arrow holds a menu under it. A menu under another menu is called a submenu. To use such a menu, position the mouse on it to display its submenu.

   For example, on the main menu, click Edit and position the mouse on Flip

Children.

8. To dismiss the menu, click Edit.

This book uses the -> arrow for the menu requests. From now on, in this book,	
Request	Means
Edit -> Copy	Click Edit then click Copy
View -> Toolbars -> Custom Click View	position the mouse on Toolbars, and then click Custom

9. Notice that, on the main menu (and any menu), there is one letter underlined on each word. Examples are F in File, E in Edit, etc. The underlined letter is called an access key. It allows you to access the same menu item using the keyboard. In order to use an access key, the menu should have focus first. The menu is given focus by pressing either the Alt or the F10 keys.
   To see an example, press Alt.
10. Notice that one of the items on the menu, namely File, has its borders raised. This means that the menu has focus.
11. Press p and notice that the Project menu is expanded.
12. When the menu has focus and you want to dismiss it, press Esc.
    For example, press Esc.
13. Notice that the Project menu has collapsed but the menu still has focus.
14. Press f then press o. Notice that the Open dialog displays.
15. On most or all dialog boxes that have either an OK, Open, or Save buttons, when you press Enter, the OK, Open, or Save button is activated. On the other hand, most of those dialog boxes also have a Cancel button (some others have a Close button instead of Cancel). You can dismiss those dialogs by clicking Cancel or pressing Esc.
    As an example, press Esc to dismiss the Open dialog.
16. On some menu items, there is a key or a combination of keys we call a shortcut. This key or this combination allows you to perform the same action on that menu using the keyboard.

    If the shortcut is made of one key only, you can just press it. If the shortcut is made of two keys, press and hold the first one, while you are holding the first, press the second key once and release the first key. Some shortcuts are a combination of three keys.

    To apply an example, press and hold Ctrl, then press S, and release Ctrl.
17. Notice that the Save As dialog box opens. To dismiss it, press Esc.

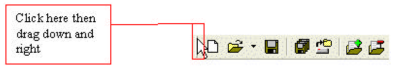
From now on, in this book,	
Press	Means
T	Press the T key
Alt, G	Press and release Alt. Then press G
Ctrl + H	Press and hold Ctrl. While you are still holding Ctrl, press H once. Then release Ctrl
Ctrl + Shift + E	Press and hold Ctrl. Then press and hold Shift. Then press E once. Release Ctrl and Shift

The Toolbars

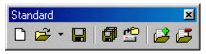
A toolbar is an object made of buttons. These buttons provide the same features you would get from the (main) menu, only faster. Under the menu, the IDE is equipped with a lot of toolbars. For example, to create a new project, you could click File -> New… on the main menu, but a toolbar equipped with the New button allows you to proceed a little faster.

By default, Bcb displays or starts with 6 toolbars. Every toolbar has a name. One way you can find out the name of a toolbar is to click and hold the mouse on its gripper bar and drag it away from its position

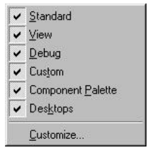
1. Drag the grippers of one toolbar down and right:



2. Notice that the toolbar has moved:

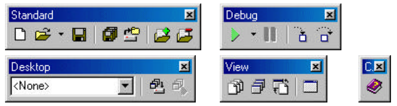


3. Borland's toolbars can be positioned anywhere on the screen. To position the toolbar back or to somewhere else, drag its title bar to its previous location.
4. You can get a list of the toolbars that are currently displaying if you right-click any button on any toolbar or menu.
   For example, right-click a toolbar and notice the list of toolbars:



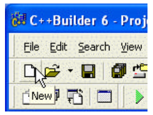
5. To dismiss the menu, press Esc.

   In this book, every toolbar is referred to by its name:



6. A toolbar is equipped with buttons that could be unfamiliar. Just looking at one is not obvious. The solution into knowing what a button is used for is to position the mouse on top of it. A tool tip that Borland calls a hint will come up and display for a few seconds.

   As an example, position the mouse (do not click) on the button that looks like a piece of paper bent on its top right corner and see the hint:



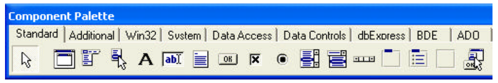
7. Without clicking, move the mouse to another button and to other buttons. Notice that some button's hints also display a shortcut that would lead to

the same action.

8. To use a toolbar's button, you click it. For example, click the New button . Notice that the action calls the New Items dialog box.
9. Press Esc to dismiss the New Items dialog box.
10. Some buttons present an arrow on their right side. This arrow represents a menu.
 To see an example, position the mouse on the Open button . Click the arrow that appears and observe the menu.
11. Press Esc to dismiss the menu.

The Component Palette

On the right side of the toolbars, there is a bar with multiple tabs; this is called the Component Palette.



The Component Palette, like a toolbar, can be moved to any location on the screen; but it is a good idea, if possible, to always keep it at its default location. The Component Palette is made of objects categorized in different tabs.

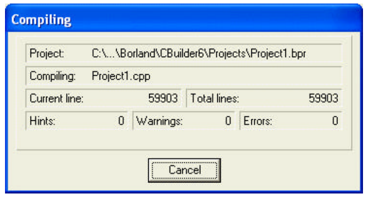
The Starting Form

Under the Component Palette, there is a gray box called a form. The form is an object that allows you to create an application that would allow the user of your program to interact with the computer. When you start Borland C++ Builder, it creates a readily available form for you. This form, although not highly functional, together with the project, provides a complete application.

An application can consist of one form or as many forms as necessary.

Notice that the current form has its own application icon, a title bar, the system buttons, and it has a body, which is the area with grids.

1. To see what the current application looks like, on the main menu, click Run -> Run.
2. Notice that the Compiling dialog box displays, showing the evolution of creating the application:



Project:	C:\...\Borland\CBuilder6\Projects\Project1.bpr				
Compiling:	Project1.cpp				
Current line:	59903	Total lines:	59903		
Hints:	0	Warnings:	0	Errors:	0

3. Once the project is ready, it displays the ready form.
 To close the form, click its close button .
4. A form is made of two parts, its "physical" part called the form and its code section called a unit.

To toggle between the form and its unit, press F12.

5. Notice that the form goes to the back.

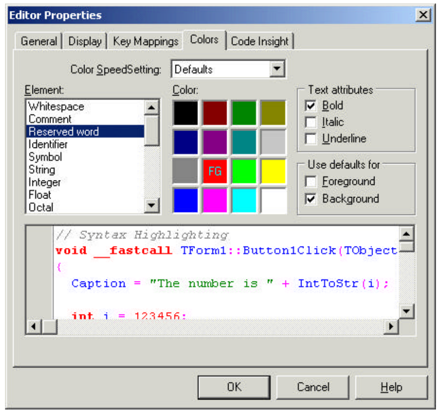
The Code Editor

One of your most regular jobs will consist of writing code that directs the computer as to what to do, when, and how. This is done in an appropriate window called the Code Editor.

The Code Editor is a full-featured text editor adapted for coding purposes. It is programmed to identify the parts of a program that are recognized by C++ or not. Therefore, the Code Editor uses different colors to differentiate the various categories of words used in a program. You can customize its display of color for each type. To do that, on the main menu, you would click Tools -> Editor Options. In the Editor Properties dialog box, click the Colors Tab.

By default, the Code Editor displays its text with a white background. Other general background colors are available from the Color SpeedSetting combo box. For example, if you select Classic, you would have a navy background.

The items that display in the Code Editor are organized by categories, and each category can have its own color. The Editor Properties provides 16 fixed colors. To change the color for a category, click it in the Element list box. To change its color, choose one from the Color panel. To change its style, use the check boxes in the Text Attributes section. The color you select for text of the category is referred to as the foreground color. You can specify a color that would permanently highlight words of that category. This is done with the Background check box:



Color SpeedSetting: Defaults

Text attributes: [✓ Bold] Italic Underline

Use defaults for: Foreground [✓ Background]

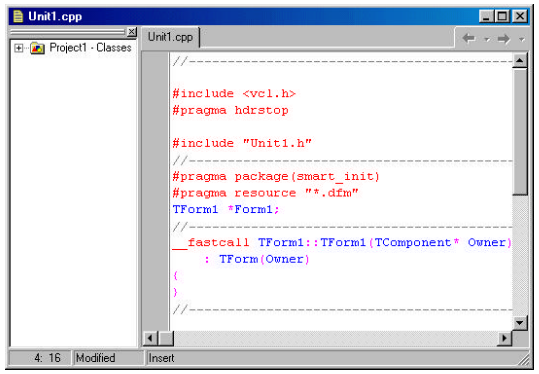
```
// Syntax Highlighting
void __fastcall TForm1::Button1Click(TObject
{
  Caption = "The number is " + IntToStr(i);

  int i = 123456;
```

While configuring the colors, you can get a preview in the memo box of the lower or lower-right section of the dialog box.

The Code Editor manages your jobs by organizing files into property sheets (also called tabs). If your project contains more than one file, you can click the desired tab to access one of the files. The basic building block of a program is called a C++ file, and Borland calls it a Unit. Whenever you start Bcb, C++ Builder creates a starting project that has a C++ file called Unit1 while the project is called Project1. Eventually, you will change these names to those you like. A typical code of a form, such as the one we have now, is built from at least two files: a header file and a source file. The file displaying now is called the source file; it gives direct instructions to the computer as to what to do on the form and why. The foundation of this source file (which is also the foundation of the form) is in a file called the header file. By default, Bcb does not display this file at startup; you have to request it.

To display the header file, you can right-click the source file and click Open Source/Header File. Indeed, this action is used to toggle both displays. Since the source and the header files go in pairs (when using classes), they hold the same name but different extensions.

Exploring the Code Editor



```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

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

1. To display the header file, press Ctrl + F6
2. Notice that the header file is called Unit1.h
3. To toggle between the form and its unit, press F12
4. To bring back the Code Editor, press F12.
5. Click Unit1.cpp.
6. Click Unit1.h
7. To bring back the form, press F12.

The Class Explorer

What is called an object in real world is also referred to as an object in C++, and an object is built using a class. To organize the objects involved in a program, C++ Builder uses a special window called the Class Explorer. As its name implies, it is used to navigate to various objects.

Whenever the Class Explorer is not displaying, to get it, on the main menu, click View -> Class Explorer

The Class Explorer is positioned on the left side (or section) of the Code Editor. It is organized in a tree view format with the name of the project as the root. To view the objects that part of a project, expand the tree.

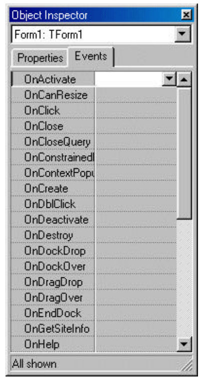
You can close or hide the Class Explorer any time and bring it back at will.

Using the Class Explorer

1. Click the + sign on the Project1 - Classes.
2. Click the + sign on TForm1
3. Since we will not use the classes for a while, to close the Class Explorer, click its Close button.
4. To bring back Class Explorer, on the main menu, click View -> Class Explorer

The Object Inspector

One of the best (if not the best) features of Borland C++ Builder (and other compilers of the company) is its ease of use and accessibility for designing objects. This is further enhanced by the presence of another special window called the Object Inspector. This is a window that lists the characteristics of the object that you are using to design an application.



The Object Inspector is useful only when designing Windows applications, which are not covered in this book.

The Object TreeView

New to Borland C++ Builder 6 (and Delphi 6), the Object TreeView is a window that displays the controls used in your application. Once again, because this book does not cover Windows applications, we will not use the Object TreeView.

Introduction to C++

Borland C++ Builder is based on the C++ language but provides a highly developed environemt for building applications. Although it looks impressive and easy, you need a (good) foundation in C++ before building such applications. The purpose of this book is to lay that foundation by learning the C++ language first.

C++ is such a huge language that part of its foundation is provided to you so that you can write your applications by adding to it. A program is made of various objects that you use to build your applications. Some of these objects have already been created and are supplied to you when using an environment such as Bcb. Although you will not see the whole functionality of these objects, you should be aware of which ones exist and how to use them.

These objects are provided in files called Header Files or libraries. By default, Borland C++ Builder's libraries are located in the C:\Program Files\Borland\CBuilder folder. Those used for C++ are installed in the C:\Program Files\Borland\CBuilderX\Include. Those you will use when creating visual applications are located in the C:\Program Files\Borland\CBuilderX\Include\Vcl folder. Although you are allowed to view these files, if you open any of them, make sure you do not touch anything; even if you see a comma that does not make sense, do not correct it.

A file that you will use as a foundation for your application is called a Header File, this is because, as "head", this file controls some aspects of your application. Therefore, you will place it at the "head" section of your program. When placing a particular file at the head of your program, you are said to "include" it. As headers, these files have an extension of .h

The most used file in C++ is called iostream.h. This file is used to display things on the screen monitor or to get things the user types using a keyboard. To use such a file, you have to "include" it using the include keyword. Here is an example:

```
include file.h
```

You must include the extension; this allows C++ to know that you are including a header file. As a rule, when including a file, you must precede the line with a # sign. Therefore, our include line would be

```
#include file.h
```

There are usually two kinds of header files you will be using in your programs: those supplied to you and those that you create. You will mostly create your own files in the same folder where you create your program. There are two ways you let C++ know where a header file is located. If the file is located in the C:\Program Files\Borland\CBuilderX\Include, which means that it was supplied to you, include it as:

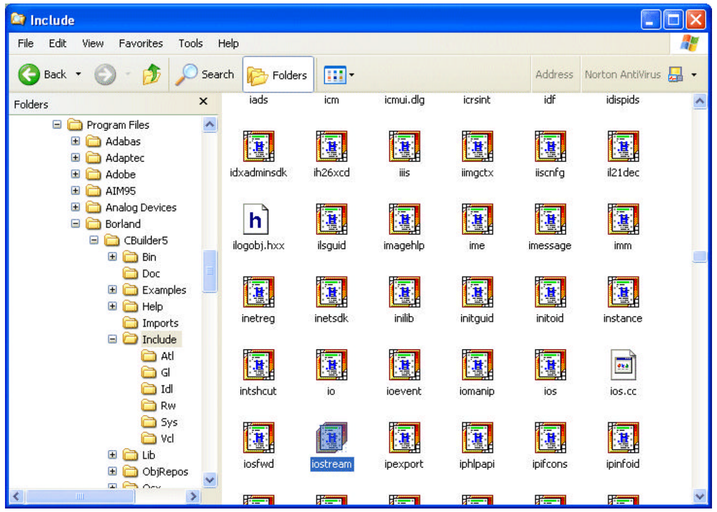
```
#include <file.h>
```

If you created your own file, which means that the file is probably located in the same folder you are creating your application, include it as follows:

```
#include "myfile.h"
```

Introducing C++

1. To view a list of the libraries that will be useful when using C++, open Windows Explorer (or Windows NT Explorer or My Computer).
2. Navigate to the C:\Program Files\Borland\CBuilderX\Include folder or where Bcb is installed.
3. Scroll down in the list and make sure you see some files such as cmath.h, conio.h, or iostream.h:


Include
File Edit View Favorites Tools Help
Back Search Folders Address Norton AntiVirus
Folders
Program Files
Adabas
Adaptec
Adobe
AIM95
Analog Devices
Borland
CBuilder5
Bin
Doc
Examples
Help
Imports
Include
Atl
Gl
Idl
Rw
Sys
Vcl
Lib
ObjRepos
iads icm icmui.dlg icrsint idf idispids
idxadminsdk ih26xcd iiis iimgctx iiscnfg il21dec
ilogobj.hxx ilsguid imagehlp ime imessage imm
inetreg inetsdk inilib initguid initoid instance
intshcut io ioevent iomanip ios ios.cc
iosfwd iostream ipexport iphlpapi ipifcons ipinfoid

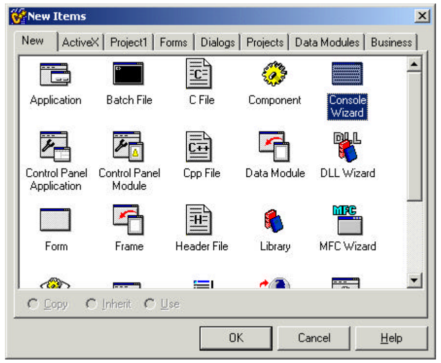
4. On the Taskbar, click Bcb to restore it.

Creating a Console Application

Borland C++ Builder allows you to create various types of applications. In this book, we will use only C++ to create console applications. A console application is one that displays its result in a DOS window. The user interacts with it by using the keyboard with very little involvement with the mouse.

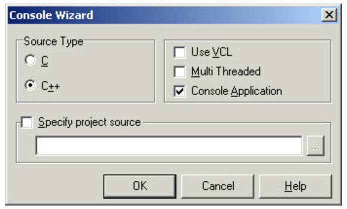
Creating a Console Application

1. On the main menu of Bcb, click File -> New... If you are using Borland C++ Builder 6, click File -> New -> Other...



2. On the New Items dialog, click Console Wizard
3. Click OK.
4. In the Console Wizard dialog, click the C++ radio button and make sure that only the Console Application check box is checked (for this

application):



Console Wizard

Source Type

( ) C

(•) C++

[ ] Use VCL

[ ] Multi Threaded

[✓] Console Application

[ ] Specify project source

5. Click OK.
6. If you receive a message box asking you to save changes made to a project, click No.
7. After a few seconds, a skeleton file is ready for you.
8. In order to make sure we start from scratch, delete everything in that file and make it empty.
9. From what we learned earlier, in the empty file, type:

   ```
   #include <iostream.h>
   ```

10. To include more than one file, you can write one under the other. For example, to include the conio.h file, change the file as follows:

    ```
    #include <iostream.h>
    #include <conio.h>
    ```

C++ Instructions

Computer programming is the art of telling the computer what to do, when to do it, and how to do it. The programmer accomplishes this by giving instructions to the computer. An example of an instruction is, "Get a number from the user", or “Display a picture on the screen". The computer carries a lot of such assignments. In C++, an assignment is called a function; it is a task you ask the computer to perform to successfully render an intended result. Like any of the objects you will be using in your programs, a function has a name. The name of a function is a letter or a combination of letters and digits. To differentiate the name of a function from other objects, the name of a function is followed by parentheses. As an assignment, a function is made of a set of instructions. These instructions are listed in a group of lines called the body of the function. The body of a function starts with an opening curly bracket "{" and ends with a closing curly bracket "}"; everything in between is part of the function. Therefore, to use a function in your program, its syntax is:

```
FunctionName() {}
```

Actually, a function is a little more than that, but for now, this is enough for us.

The most used function in C++ is called main. The main() function can take various forms.

```
main() {}
```

A C++ file is made of statements; these are sequences of actions you ask C++ to perform while the program is running. You can write such a statement on one line, or you can span more than one line. By default, when you start a

new line, C++ believes that you are starting a new statement. To let C++ know when a statement ends, you write a semi-colon at the end of the line. For example, to write one statement, you could use:

```
Here-Is-A-C++-Statement;
```

To span more than one line, type a semi-colon only when the statement is over. Here is an example:

```
A-Long-Statement-From-C++-That-Spans-
More-Than-One-Line;
```

Introduction to Functions

1. From what we have learned so far, change the content of your file as follows:

   ```
   #include <iostream.h>
   #include <conio.h>
   main(){}
   ```

2. You can include one function inside of another. As an example, change the main() function as follows:

   ```
   main(){getchar();}
   ```

Executing Programs

A program would not mean much unless it accomplishes the desired purpose. To examine how your development is proceeding, as a beginning programmer, you should regularly ask C++ to show you the result.

The (small) program we have written is really plain English, something we can read and analyze. Unfortunately, the computer does not understand what it means; this is because the computer "speaks" its own language called the Machine Language. For the computer to understand what your program means, you need an intermediary program that can translate your English to machine language and vice versa. This program is called a compiler, and it is supplied to you. The process of executing a program is done in various steps that Borland C++ Builder can resume as one.

There are three ways you can execute a program in Borland C++ Builder. To execute a program, you can press F9, you can also use the main menu where you would click Run ª Run. On the toolbar, you can also click the Run button

Executing a Program

1. On the Debug toolbar, click the Run button ▶.
   As you see, the program does not do much because we did not write a formal assignment.
2. To close the DOS window, press Enter on your keyboard.

C++' cout Operator

Although it worked fine, the program we have just used lacks many things. You should make your programs easy to read and navigate; this is accomplished by writing each statement on its own line. For example, the above program can be re-written as follows:

```
#include <iostream.h>
#include <conio.h>
main()
{
    getchar();
}
```

The iostream.h library contains a special operator used to display something on the screen.

cout is a class and not an operator. For now, we will call it an operator.

This is done with the cout operator (pronounce "see-out"). To use this operator, type it followed by two "less than" signs "<<", the statement you want to display, and end the line with a semi-colon. An example would be;

cout << Line-Of-Statement;

There are mainly two kinds of items you display with a cout operator: expressions or strings. An expression is a character, a symbol, a word or a combination of those, usually from performing an operation or an assignment. There are specific rules we will learn and follow when displaying an expression. A string is a character, a symbol, a word or a group of words that you ask the compiler to display "as is". To display a string, you enclose it between a pair of double-quotes. For example, to display C++ Programming, you would write:

cout << "C++ Programming";

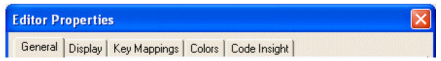
You can also display a long string and span more than one line. In this case, the cout would detect the end of the string when it finds a semi colon after the second double-quotes. There are two methods you can follow. To display a long string, after the cout operator and its <<, start the string with double-quotes; include each line in double-quotes; the compiler will append each line to the next. Here is an example:

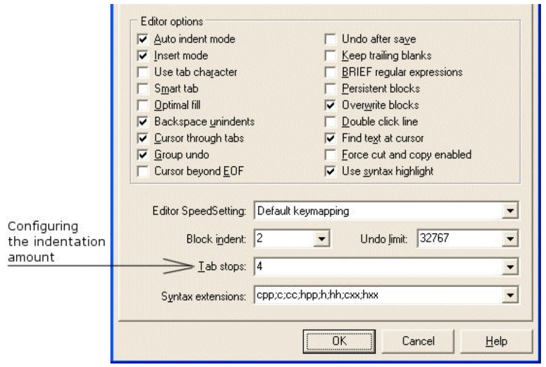
```
cout << "In 1873 Mark Twain and Charles Dudley Warner created an enduring "
"label for their times when they collaborated on a novel entitled "
"The Gilded Age. Ref: America A Narrative History.";
```

To display sections of strings, you can use one cout but each line should start with its own << operator. Here is an example:

```
cout << "The White House "
<< "Pennsylvania Avenue";
```

Indentation is another feature that makes your program easy to read. Indentation is a technique of grouping lines of code by category. To effectively use indentation, identify some special symbols in your program. For example, we saw that the body of a function starts with an opening curly bracket "{" and ends with a closing one "}". To delimit the items that are inside of a function, you should indent them by two empty spaces or one tab. When inside of a function, if you press Tab, C++ Bulder refers to the indentation amount specified in the Editor Properties. By default, C++ Builder sets a Tab press to 8 characters, which usually seems too much. Most programmers, including me, want a Tab to be equivalent to 4 characters. To change the indentation amount when pressing Tab, open the Editor Properties dialog box (Tools -> Editor Options...) and click the General property sheet. In the Tab Stops combo box, specify the amount you want and click OK:




Editor options

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

Editor SpeedSetting: Default keymapping
Block indent: 2 Undo limit: 32767
Tab stops: 4
Syntax extensions: cpp;c;cc;hpp;h;hh;cxx;hxx

Using indentation, the program could be written:

```
#include <iostream.h>
#include <conio.h>
main()
{
    getchar();
}
```

Indentation should be incremental. That is, when a line of code appears to be a child of the previous line, the new line should be indented. The address lines we saw earlier can be written as:

```
cout << "The White House "
     << "Pennsylvania Avenue";
```

The cout Extraction Operator

1. Change the content of the file as follows:

```
#include <iostream.h>
#include <conio.h>

main()
{
getchar();
}
```

2. To use the cout operator, change the main() function as follows:

```
#include <iostream.h>
#include <conio.h>

main()
{
cout << "Press any key to continue...";
getchar();
}
```

3. To execute and test your program, on the main menu, click Run -> Run
4. To close the DOS window, press Enter, and return to Bcb.

C++ Comments

Comments in a C++ file serve as documentation entities that help you and other people to read your program. Comments are lines of text that you can write in plain English because the compiler skips them completely.

There are two styles of comments in C++. To write a comment on one line, start the line with two forward slashes. Here is an example of a comment:

// The compiler doesn't read this line.

To span a comment on more than one line, you can start each line with //. An alternative is to include the comment section between /* and */. Here is an example:

/* The compiler will ignore this whole section and will jump over it. It is used to span more than one line of comments*/

❖ Using Comments

1. Change the content of the file as follows:

```
// Introductory lesson to C++
#include <iostream.h>

/* This is the main function.
   It is included on every C++ program.
   It is the central point of a C++ application.*/

main()
{
    cout << "Press any key to continue...";
    getchar();
}
// -------------------------------------------
```

2. Execute the program to see the result.

Escape Sequences

An escape sequence is a character that holds a special meaning to the compiler such as switching to the next line or adding a tab indentation. Although an escape sequence is a character by definition, it is created using the backslash "\" followed by a specific character. For example, the escape sequence \n asks the compiler to add or create a new line, also referred to as Carriage Return or "Carriage Return – Line Feed". The new line can also be created with a stand-alone endl. Here is an example:

cout << endl;

To use an escape sequence by itself, you can include between single quotes: '\n' or double quotes: "\n". Because escape sequences are specially recognized by the compiler, they can be included in a cout string. Since the compiler will still read a string, if it finds an escape sequence, it acts accordingly. Here is a list of escape sequences and their meanings:

Escape	Meaning
\a	Produces a bell sound
\b	Backspace
\f	Formfeed
\n	Newline – Linefeed
\r	Carriage return
\t	Horizontal Tab
\V	Vertical Tab
\\	Backslash
\'	Single Quote or Apostrophe
\"	Double-Quote
\?	Question Mark

❖ Using Escape Sequences

1. To implement examples of using escape sequences, change your program as follows:

```
// Introductory lesson to C++
#include <iostream.h>

/* This is the main function.
It is included on every C++ program.
It is the central point of a C++ application.*/

main()
{
    cout << '\t' << "- Borland C++ Builder -\a" << endl;
    cout << "This will end our session today" << "\n";

    cout << "\n\nPress any key to continue...";
    getchar();
}
```

2. To test the program, press F9.
3. After using the program, return to Bcb.

Saving a Project

A program in Borland C++ Builder is called a project. As an application, it is saved in a few steps. To save a project, on the Standard toolbar, you can click the Save All button.

❖ Saving a Project

1. On the Standard toolbar, click Save All .
2. Type Main to replace the name of the Unit1.
3. Click the arrow of the Save In combo box and click the My Documents.
4. Click the Create New Folder button .
5. Type Exercise01 and press Enter.
6. Double-click Exercise01 to display it in the Save In combo box.
7. Click Save
8. Type Exercise to replace the name of the project.
9. Click Save.

Borland C++ Builder Help

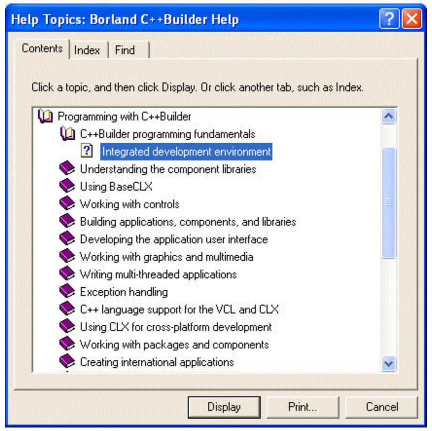
There are two main sources of help available for Borland C++ Builder. The first source of help is provided with the compiler, this is from the Borland Company. This help is referred to as Online Help. In addition to the Online Help, Borland C++ Builder usually ships with a book titled Programmer Guide. Fortunately, you have access to this help even if you are not designing an application. Everything considered, this is the closest and the highest documentation that the compiler provides.

To access C++ Builder help, on the task bar, click Start -> Programs -> Borland C++ Builder -> Help, and make your choice from the categories.

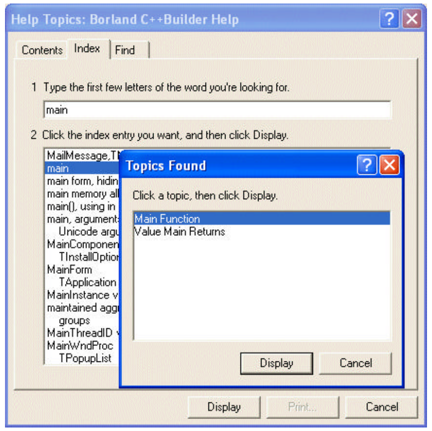
Another place you can find information is on the Internet. Fortunately, most of that help is free. Unfortunately, many of these sites tend to treat C++ Builder and Delphi as one entity, which is not the case.

Online Help

1. On the main menu, click Help -> C++ Builder Help.
2. In the Contents tab, double-click Programming With C++ Builder to expand it.
3. Double-click C++ Builder Programming Fundamentals
4. Double-click Integrated Development Environment:


Help Topics: Borland C++Builder Help
Contents
Index
Find
Click a topic, and then click Display. Or click another tab, such as Index.
Programming with C++Builder
C++Builder programming fundamentals
Integrated development environment
Understanding the component libraries
Using BaseCLX
Working with controls
Building applications, components, and libraries
Developing the application user interface
Working with graphics and multimedia
Writing multi-threaded applications
Exception handling
C++ language support for the VCL and CLX
Using CLX for cross-platform development
Working with packages and components
Creating international applications
Display
Print...
Cancel

5. Notice that the help window displays, click the advance button .
6. After reading, click Help Topics
7. Click the Index tab.
8. Type main and press Enter
9. Since the word main is addressed in many articles, notice that a second dialog displays:



10. Click Main Function and click Display
11. Click the Arguments To Main link.
12. After reading the text, on top of the window, click Example
13. After reading the example, click Back.
14. To close the Help system, click its Close button .

Internet Help

Perform the following exercise if you have access to the Internet.

1. Log on to http://www.borland.com
2. Look for and click C++ Builder.
3. After finding some information and reading, change the address in the Address box to http://msdn.microsoft.com and press Enter.
4. Position the mouse on Libraries and click Developer

Next