

2

How to compile and test a program with Net Express 4.0

In this chapter, you'll learn how to edit, compile, test, and debug a program when you're using the University Edition of Net Express 4.0. When you finish this chapter, you'll be able to develop programs on your own PC using that compiler.

In addition, this chapter will give you a good background for learning how to develop programs on other platforms or with other compilers. Even if you use a different compiler on the job or in your school lab, though, you can use the University Edition of Net Express 4.0 to learn on your own PC. It is a relatively inexpensive product that is a terrific training tool.

If you do use a different compiler on the job or at your school, you will of course have to learn how to compile and test programs with that compiler too. If, for example, you're going to develop programs for an IBM mainframe, chapter 18 shows you how to compile and test programs on that platform. You can read that chapter any time after you complete this one.

Introduction to the Net Express 4.0 IDE	2
The Integrated Development Environment of Net Express 4.0	2
How a production program is compiled, link edited, and executed	4
How to enter and edit a program	6
How to open an existing project	6
How the project files and folders can be organized	6
How to start a new project	8
How to add files to a project	10
How to set the default folder	12
How to enter and edit source code	14
How to find and replace text	16
How to print the source code for a program	16
How to compile and test a program	18
How to compile a program	18
How to correct compile-time errors	18
How to test a program	20
How to correct run-time errors	22
How to use the debugging features	24
How to display and modify the values of variables	24
How to use breakpoints	26
Perspective	30

Introduction to the Net Express 4.0 IDE

To develop a program using the University Edition of Net Express 4.0, you use its *Integrated Development Environment*, or *IDE*. The IDE for this edition of Net Express 4.0 works the same way as the full version of Net Express 4.0. It also works essentially the same way as the IDE for Micro Focus Mainframe Express, which is the version that programmers use for developing COBOL programs for IBM mainframes. As a result, you'll be able to transfer the skills you learn in this chapter to both Mainframe Express and the full edition of Net Express.

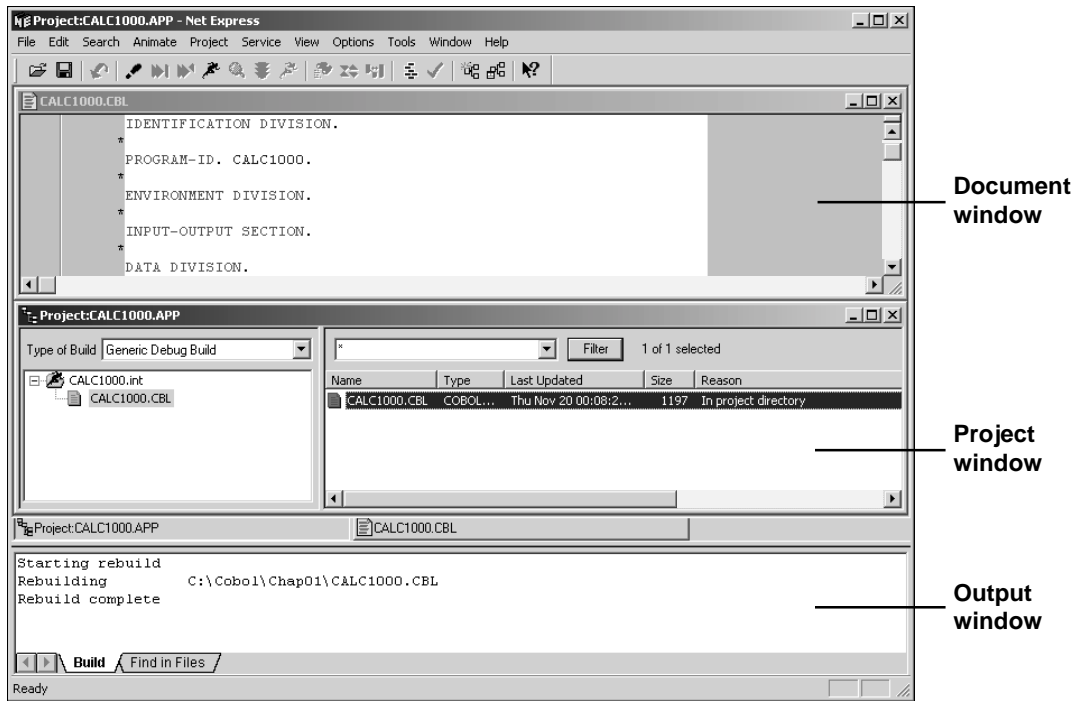
The Integrated Development Environment of Net Express 4.0

When you develop programs with the IDE for Net Express 4.0, you work with the three main windows shown in figure 2-1. The *project window* shows all of the files that the *project* contains. The *document window* shows the COBOL code for a COBOL file. And the *output window* shows various messages that are generated as you develop a program.

For most of the programs that you develop for this book, the project will contain one COBOL file with the extension CBL and a second intermediate file with the same name as the COBOL file but with INT as the extension. The CBL file contains the COBOL *source code* for the program, and the INT file contains *intermediate code* that is created when you *compile* the program. The intermediate code is used when you *execute* (or *run*) the program.

In the rest of this chapter, you'll learn how to use this IDE to start a project, enter the source code for a program, build the program, and execute the program. But before you learn those skills, you should understand how developing programs with the University Edition of Net Express differs from developing production programs.

The three main windows of the Net Express IDE in tiled view



Description

- When you develop a COBOL program with Net Express 4.0, you use the three main windows of the Net Express *Integrated Development Environment* (or *IDE*).
- To start your work with an existing application, you open the *project window* for the *project*. Within this window, you can see all of the COBOL files (extension `cbl`) that the project contains. For most of the applications in this book, there will be just one program file in each project.
- To enter and edit a COBOL file, you can double-click on its file name in the project window. This opens the *document window*. The code that you enter can be referred to as *source code*, and the file that's created can be referred to as a *source file*.
- To *compile* the program, you can click on the Build button in the toolbar. This creates an *intermediate file* with the same name as the program, but with `int` as the extension.
- To *execute* (or *run*) a program, you can click on the Run button in the toolbar. This executes the intermediate file for the program under the control of Net Express.
- If any errors are detected as the program is compiled or run, messages are displayed in the Build tab of the *output window*.

Figure 2-1 The Integrated Development Environment of Net Express

How a production program is compiled, link edited, and executed

When you use the University Edition of Net Express for developing student programs, Net Express controls the execution of the programs that you develop. In contrast, when you develop *production programs* that are used on the job, the programs run by themselves .

To illustrate the differences, figure 2-2 shows how a production program is developed. After the programmer uses an editor to create the source program, the programmer initiates a three-step procedure that's run by the computer. As you can see, the output of the second step is an *executable* that is run by itself in the third step. In contrast, the University Edition of Net Express doesn't produce an executable. Instead, Net Express uses an intermediate file to run the program.

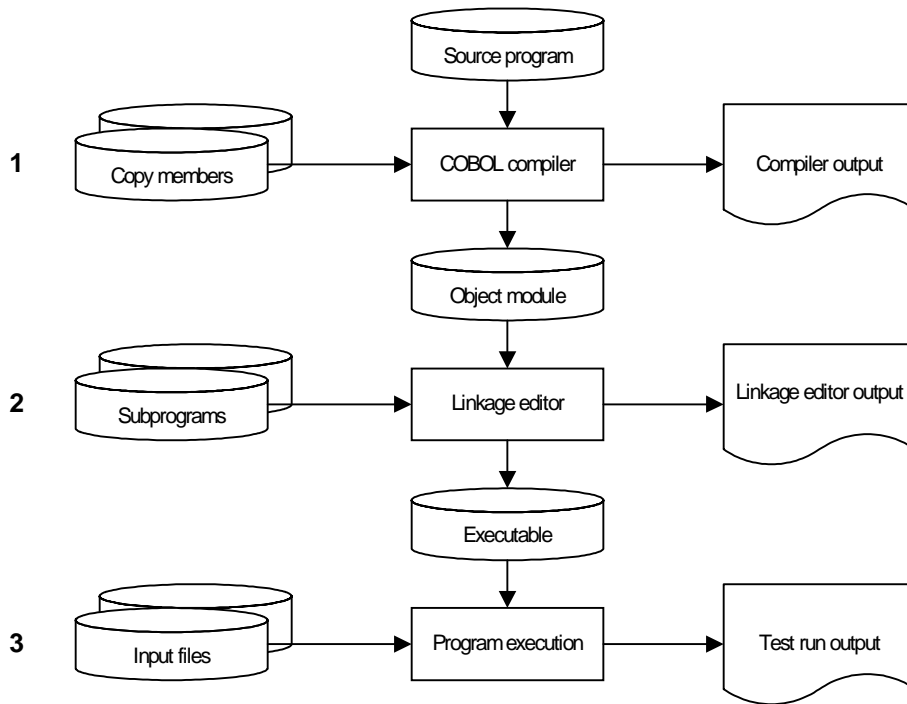
In step 1 in this figure, the COBOL compiler compiles the source program into an *object module*. If the source program uses any Copy statements (see chapter 11), the compiler copies the source code from the related copy members into the program as part of this process. The compiler also produces output like a compiler listing and a list of the compile-time errors (often called *diagnostics*). Sometimes, this output is printed, but it is often reviewed on the monitor or screen without ever printing it.

In step 2, a program called the *linkage editor* combines the object module with any subprograms that the program requires into an executable. This is called *link editing* (or *linking*). On most platforms, one or more system subprograms are required in this step. These subprograms do some of the specific types of processing that the program requires. In addition, the linkage editor can link the object module with user subprograms (see chapter 11). Although the linkage editor also produces some printed output in this step, programmers rarely need to refer to it.

In step 3, the executable version of your program is executed. This is the test run of your program. As a result, the program gets the input that it requires including keyboard or disk data, and the program produces the output it requires including display, disk, and printer data.

With minor variations in the terminology, this is the way a production program is prepared on all platforms: compile, link edit, and test. If you want to learn more about how this is done on an IBM mainframe, please read chapter 18.

The three-step procedure that's done by the computer



Description

- Before this procedure can be run by the computer, the programmer uses an editor to enter the source program into a source file. Then, the programmer initiates this procedure.
- In step 1, the *COBOL compiler* compiles the source program into an *object module*. In step 2, the *linkage editor link edits* the object module into an *executable*. In step 3, the executable is run so the programmer can see whether the program works.
- During step 1, the COBOL compiler inserts the source code that's in the copy members that are referred to by any Copy statements in the program (see chapter 11). The compiler also produces output like a compiler listing and a list of compile-time errors.
- During step 2, the linkage editor link edits the object module with any system subprograms or user subprograms that it needs (see chapter 11). The linkage editor also produces linkage editor output.
- Step 3 is the test run for the program. It gets whatever input the program specifies and produces whatever output the program specifies.

Figure 2-2 How a production program is compiled, link edited, and tested

How to enter and edit a program

Now that you have a general idea of how you develop a COBOL program, you're ready to learn the specific skills for developing your own programs with the University Edition of Net Express. You'll start by learning how to enter and edit a COBOL program.

How to open an existing project

If you want to continue work on an existing project, you can use the procedures shown in figure 2-3 to open the project. If the Welcome dialog box is displayed when you start Net Express, you can click on the Continue button to open the last project you were working on. Otherwise, you can use the Open command in the File menu (File→Open) to display the Open dialog box. Then, you can use normal Windows techniques to find and open the project file that you want. Note that this type of file has APP as the extension.

When a project is opened, only the project and output windows are displayed in the IDE. Then, you can open the document window for a COBOL file by double-clicking on it.

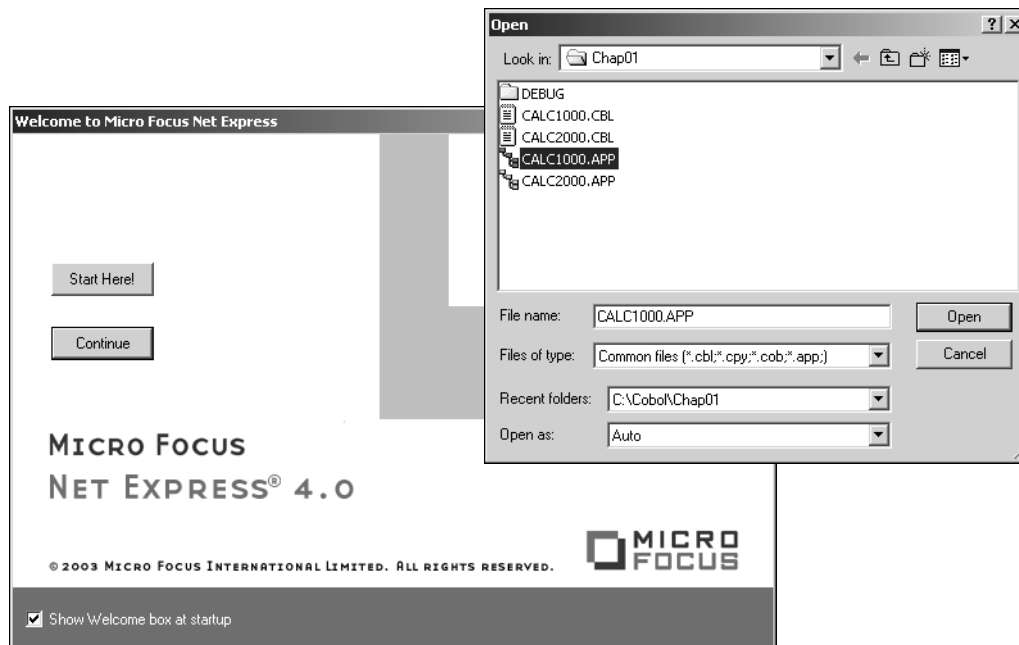
How the project files and folders can be organized

When you use Net Express, each project contains one or more COBOL files. To keep these files organized, both the project file (app) and the COBOL files (cbl) should be stored in the same folder. However, more than one set of project and COBOL files can be stored in the same folder because each project file keeps track of the COBOL files that the project contains.

With that in mind, there are two logical ways to organize the files and folders for the programs that you create. For student programs, it's reasonable to store more than one project in the same folder. In figure 2-3 for example, the Open dialog box shows that two projects (CALC1000 and CALC2000) are stored in a folder named Chap01. In other words, all the programs for a single chapter are stored in a single folder.

For production programs, though, it's common to store just one project in each folder. That makes it easier to find the projects that you're looking for. This is especially useful for projects that contain more than one COBOL file.

The Welcome and Open dialog boxes



Three ways to open an existing project

- Click on the Continue button in the Welcome dialog box. That will open the project that you were working on when you ended your last Net Express session.
- Use the File→Recent Projects command to open a recent project.
- Use the File→Open command to display the Open dialog box, and use normal Windows techniques to find the folder that the file is in. Then, double-click on the project (app) file that you want to open, or select that file and click on the Open button.

Two ways that the folders and files for an application can be organized

- For production projects that contain more than one file, it is best to put each project in its own folder. If you use this approach, the CALC1000 project will be in one folder and the CALC2000 project will be in another.
- Because it is the app file that keeps track of all of the files for a project, you can also keep more than one project in a single folder. This is usually acceptable for student projects. If you use this approach, you can keep two or more projects like CALC1000 and CALC2000 in a single folder as shown above.

Figure 2-3 How to open an existing project

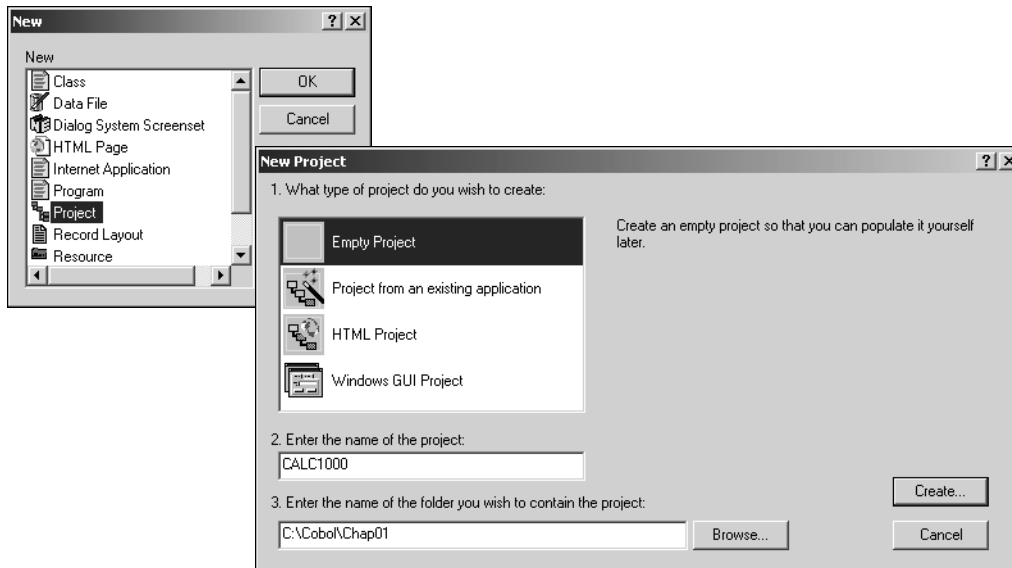
How to start a new project

Before you start a new program, you need to start a new project. To do that, you use the procedure in figure 2-4. In the New Project dialog box, you provide the name of the project and the name of the folder that you want the new project stored in. If that folder doesn't exist, Net Express will create it for you. This makes it easy for you to create the folders that you want to use for your projects.

When you complete this procedure, a new project (.app) file is created and the project window is opened. Then, you can add a COBOL file or a copy of a COBOL file to the project as shown in the next figure.

Another way to start a new project and add an existing file to it is to use the Project from an existing application option in the New Project dialog box. Then, when you click on the Create button, a Wizard starts that lets you add one or more existing files to the project. When you complete the Wizard, the project window is opened, and it shows the files that have been added to it.

The New and New Project dialog boxes



How to start a new project

1. Use the File→New command to open the New dialog box. Then, select Project and click the OK button to display the New Project dialog box.
2. In the New Project dialog box, select Empty Project, enter the name of the project, and enter the name of the folder that you want the project saved in. Then, click on the Create button.
3. This creates a new app file for the project and opens the project window. If the folder specified in the New Project dialog box doesn't exist, Net Express asks whether you want the folder created.

Description

- If you're creating a new program, it's usually best to start an empty project and then add files to it as shown in the next figure.
- If you want to create a new project for an existing program, you can select the Project from an existing application option in the New Project dialog box. This starts a wizard that steps you through the process of adding existing files to the project.

Figure 2-4 How to start a new project

How to add files to a project

After you create a new project, you need to add one or more COBOL files to the project. To do that, you can use the procedures shown in figure 2-5.

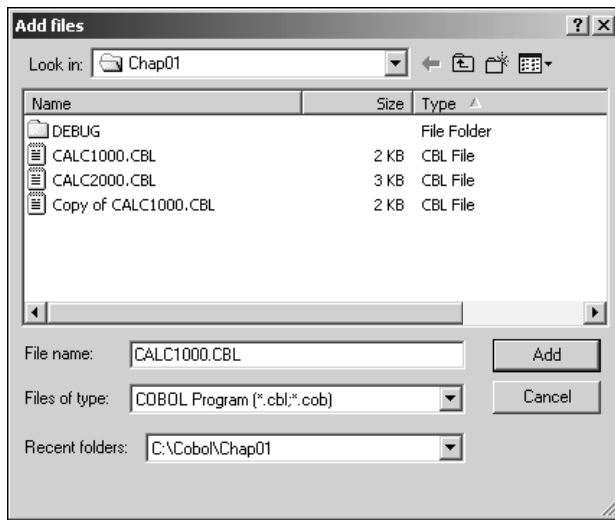
In the Add files dialog box, if you specify a file name that doesn't already exist, Net Express will create an empty file with that name. Then, you can enter COBOL source code into that file. Because every COBOL program requires some of the same coding, though, you should rarely, if ever, start a new program from scratch.

Instead, you should start a new program from an old program that is similar to the one you're going to develop. At the least, the old program will have the required division and section headers in it. Then, you can delete the statements you don't need, modify the statements you do need, and add the new statements that you need. Often, you can pick up dozens of statements from an old program when you start a new program this way. As a result, this is one of the keys to programmer productivity.

When you use Net Express, the best way to start a new program from an old program is to add a copy of an existing file to the new project that you've created. That can be done by using the three-step procedure in this figure. In step 1, you find the old file that you want to start the new program from and copy the file to the clipboard. In step 2, you move to the folder that the new project is stored in and paste the old program from the clipboard into that folder. In step 3, you rename the file that you've pasted with the name that you want to use for the new program.

Another alternative is to first use the Windows Explorer to copy and paste the old source file into the folder that you're going to use for the new project. Then, you can start a new project in that folder and use the Add files dialog box to add the source file to the project. Or, you can use the New Project dialog box in the previous figure to both start a project and add the copied file to the project. In that case, you use the Project from an existing application option and let the Wizard walk you through the process.

The Add files dialog box



How to open the Add files dialog box

- Right-click in the left pane of the project window to display the shortcut menu. Then, select the Add file to project command.

How to add a copy of an existing file to the project

1. Use the controls in the Add file dialog box to move to the folder that contains the file that you want to copy. Then, right-click on the file that you want to copy and select the Copy command. This copies the file to the clipboard.
2. Use the controls in the Add file dialog box to move to the folder that contains the project that you're adding the file to. Then, right-click in the file area and select the Paste command. This copies the file from the clipboard to the folder.
3. If necessary, rename the file that you've pasted into the folder. One way to do that is right-click on the file name and select the Rename command. Then, select that file and click on the Add button.

Description

- To work productively, you should start each new program from a copy of an old program that is similar to the one you're developing.
- If you add an existing file to a project, the project points to that file no matter what folder it is in. It doesn't automatically make a copy of it and put it in the same folder as the project file. As a result, the programmer is responsible for copying existing files, pasting them into the right folders, and renaming them.

Figure 2-5 How add files to a project

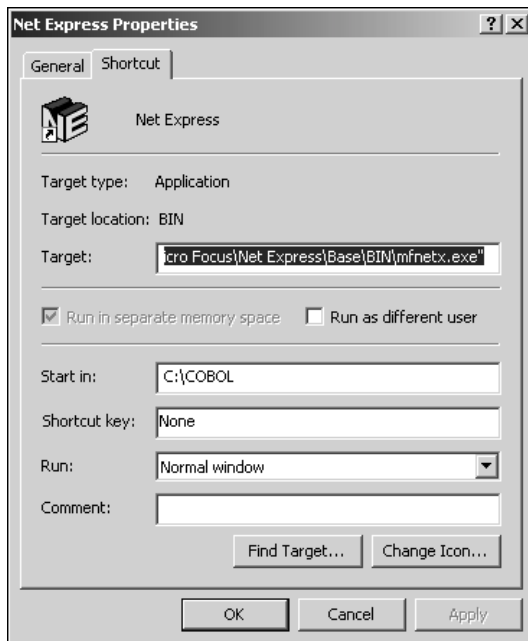
How to set the default folder

To make it easier to find and open the files that you're working with, you should change the default folder to one that you commonly use. If, for example, you commonly work with folders and files that are contained within the `c:\cobol` folder, you can change the default to that folder.

As you can see in figure 2-6, you use the Net Express Properties dialog box to change the default folder. To display this dialog box, you click on the Start button in the Windows taskbar and continue until you find the Net Express program. Then, you right-click on it to display its shortcut menu, and choose the Properties command.

In the Properties dialog box, you enter the path for the folder that you want to use as the default in the Start in box. In this figure, the default folder is being changed to `C:\COBOL`. Since the path isn't case sensitive, you don't have to worry about the capitalization.

The Properties dialog box for Net Express



How to display the Net Express Properties dialog box

- Click on the Start button in the Windows taskbar, click on Programs, click on Micro Focus Net Express 4.0 UE, and right-click on Net Express to display the shortcut menu. Then, click on the Properties command.

How to change the default folder

- Change the Start in folder in the Properties dialog box to the folder that you want as the default folder. This change will take effect the next time you start Net Express.

Description

- After you install Net Express, the default folder is set to a Demo folder that contains some demonstration programs. For efficiency, though, you should change the default to the top folder for the projects that you're developing.

How to enter and edit source code

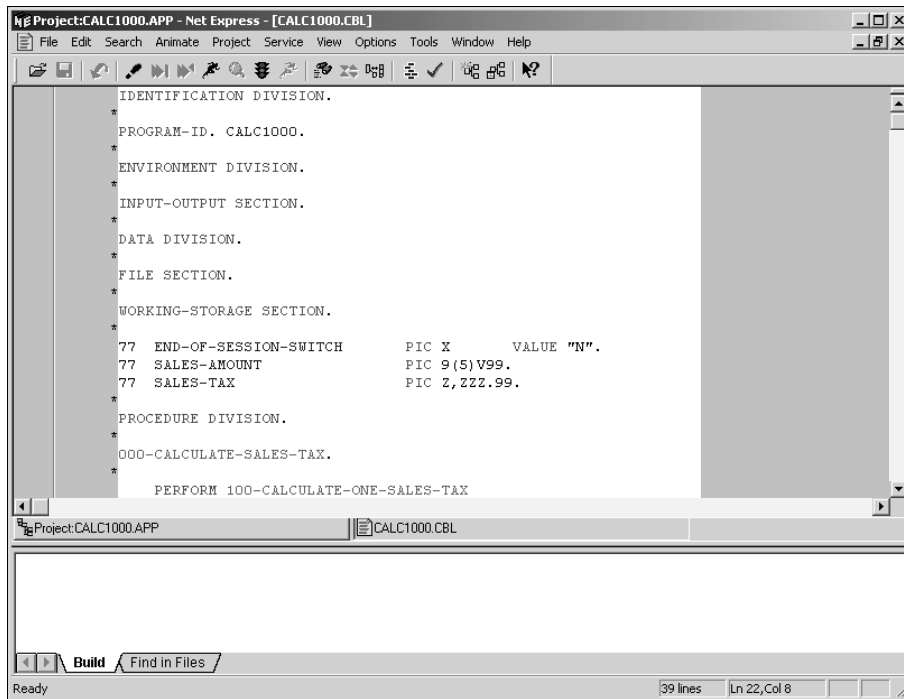
Once you've created a new project file and new program file, you can double-click on the COBOL file in the project window to open the document window for the file. This window is shown in figure 2-7. Notice that the left and right margins are set the way you want them for a COBOL program because that's how the *COBOL profile* for Net Express is set up. In addition, this profile provides for tab stops every fourth position so it's easy to align and indent your code.

When you open the document window, you usually have to adjust the Net Express windows so they're easier to work with. To do that you can use normal Windows techniques like clicking on the Maximize button, dragging the border of a window, or using the commands in the Window menu.

When a document window is displayed and the project window isn't, you can move from one window to the other using normal Windows techniques like pressing Ctrl+F6 or using the Window menu. You can also click on the tabs at the bottom of a document or project window to move from one window to another.

Because this book assumes that you are already familiar with the way Windows programs work, it doesn't present detailed operational instructions for entering and editing code. As a result, this figure just summarizes some of the typical editing operations. Later on, when you experiment with the Net Express, you'll see that entering and editing a program isn't much different from entering and editing a Word document. As a result, it won't take you long to get comfortable with this process.

The document window for a source program



Description

- Net Express uses colors in the document window to identify various elements of code. For instance, the COBOL reserved words are green, variable names are red, literals are black, procedure names are blue, comments are gray, and unidentifiable words are purple.
- The *COBOL profile* is set so the white area of the document window represents positions 8 through 72 with tab stops at every fourth position.
- To move to the next tab, press the Tab key. To move to column 7 so you can enter an asterisk into it, use the left arrow key or click on it with the mouse.
- To delete one or more lines of code, highlight them and press the Delete key. To move or copy lines of code, highlight them and use the standard Cut (Ctrl+X), Copy (Ctrl+C), and Paste (Ctrl+V) commands. To undo an operation, click on the Undo button or press Ctrl+Z.
- To change the editing defaults, you can use the Options→Edit command.
- To move from the document window to the project window or vice versa, you can click the tabs at the bottom of the current window, use the Window menu, or press Ctrl+F6.

Figure 2-7 How to enter and edit source code

How to find and replace text

As you enter and edit a COBOL program, you sometimes need to find or replace specific segments of text. To do that, you can use the procedures shown in figure 2-8. When you click on the Find Text button in the toolbar, the Find and Replace window is displayed below the output window, and you can proceed from there.

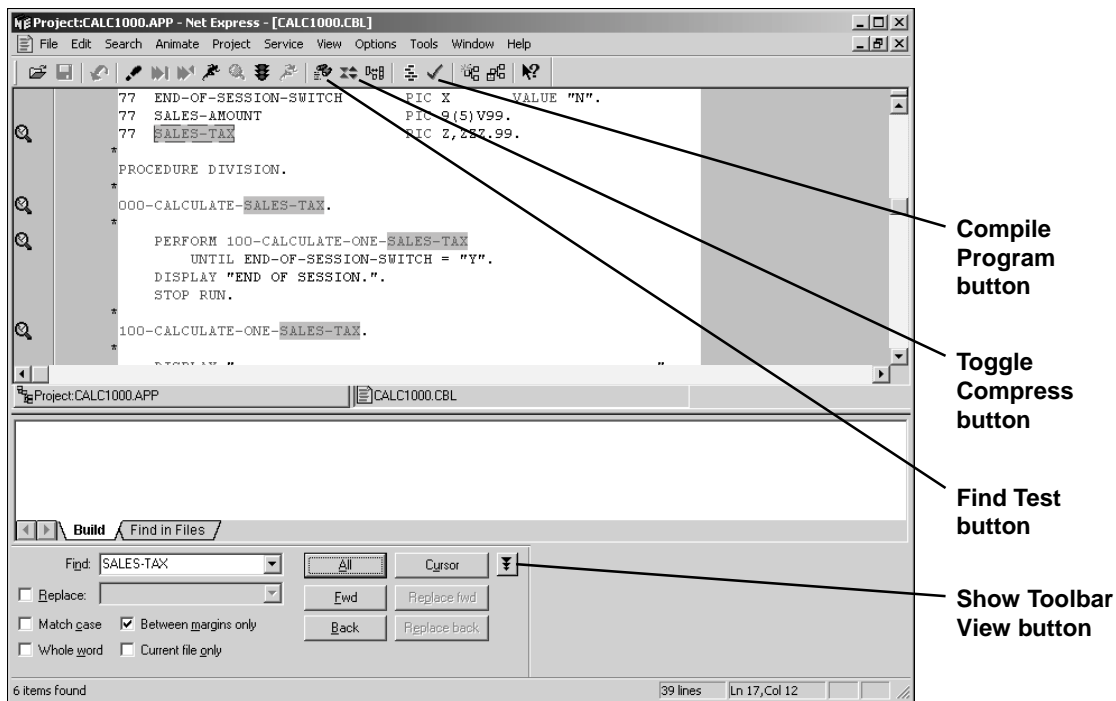
When you click on the All button to find all the occurrences of the text, the lines that contain those occurrences are tagged and highlighted as shown in this figure. Then, you can move from one *tagged line* to another. You can also *compress* all of the tagged lines so only those lines appear in the document window, after which you can *expand* the tagged lines so all of the lines appear again. The easiest way to remove all of the tags is to compile the program by clicking on the Compile Program button in the toolbar.

If you click on the Show Toolbar View button, the Find and Replace window is replaced by the Find toolbar. Then, you can use the first two buttons on this toolbar to move from one occurrence of the specified text to another. When you want to go back to the Find and Replace window, you can click on the Show Dialog View button at the right of the toolbar.

How to print the source code for a program

To print the source code for a program, you can use the Print command in the File menu just as you do with any Windows program. This can be useful when you're debugging a long program or correcting its compile-time errors. As you get used to working with programs on the screen, though, you'll find that you rarely need printed listings.

The IDE after the FindText command has been executed



How to use the FindText command to find and replace text

1. To start the operation, click on the Find Text button in the toolbar. This opens up the Find and Replace window beneath the output window.
2. In the Find and Replace window, enter the Find and Replace values and check the boxes that determine the way the command works.
3. To find and replace one occurrence at a time, click on the Fwd or Back buttons. Or, to find, replace, and tag all occurrences in the document, click on the All button.

How to move to, compress, expand, and remove tagged lines

- To move from one tagged line to another, click on the Fwd and Back buttons. Or, click the Show Toolbar View button to display the Find toolbar and click on its first two buttons.
- To compress the display so only the tagged lines are shown, click on the Toggle Compress button. To redisplay all the lines of code, click on the button again.
- To remove the tagged lines, click on the Compile Program button.

Figure 2-8 How to find and replace text

How to compile and test a program

Once you've entered the source code for a program, you're ready to compile and test it. During these steps of program development, you'll find and correct any errors in your program.

How to compile a program

Figure 2-9 shows how to compile a program. The easiest way to start the compile is to click on the Build button in the toolbar. If the project contains more than one COBOL file, this compiles all of them and checks to make sure that they will work properly together. But otherwise, it compiles just the one COBOL file in the project.

When you compile a program, Net Express checks the syntax of the code to be sure it's correct. When it's done, it lists all the errors that it finds in the output window as shown in this figure. These errors can be referred to as *compile-time errors*. In addition, Net Express marks each line of code that contains an error with an X in the document window so the error statements are easy to locate.

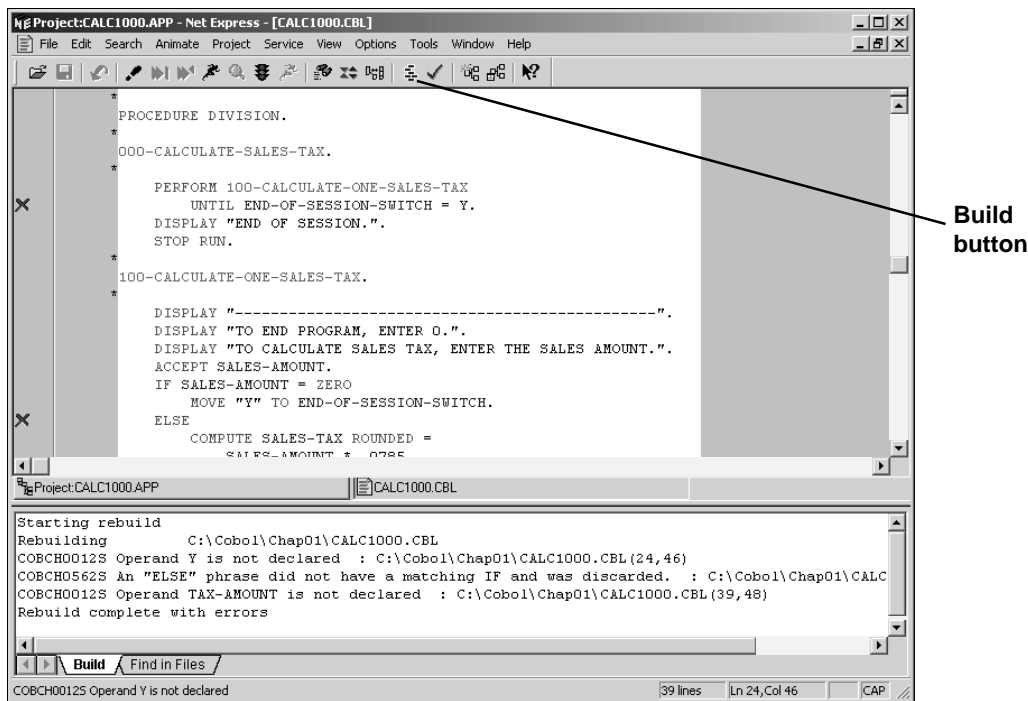
How to correct compile-time errors

Most of the time, you can find the cause of the compile-time errors by studying the error messages and the source code. If, for example, you study the messages and code in this figure, you should be able to figure out what caused the errors without much trouble. If you can't do that, though, the exercises at the end of this section will guide you through the corrections.

This figure also describes some techniques that you can use to work with the list of errors in the output window and the marked statements in the document window. Specifically, you can double-click on an error in the output window to move to the statement that's in error in the source code. You can also compress and expand the error lines by using the Toggle Compress button in the toolbar.

After you correct the errors, you need to compile the program again. Then, if additional errors are detected, you need to repeat the correction process until the program compiles without any errors. That can be referred to as a *clean compile*. When the program compiles with no errors, the compiler creates the intermediate file that is used to run the program.

The document window after a compile with errors



Description

- The best way to start a compile is to click on the Build button.
- When an error occurs during a compile, the lines of code that contain errors are preceded by X marks and a description of each error is given in the output window.
- If you double-click on an error in the message window, the cursor moves to the line of code that contains the error.
- To compress the display so only the error lines are displayed, click on the Toggle Compress button. To expand the display, click on the button again.
- If you study the error messages and the source code, you should be able to figure out what's wrong with each highlighted statement. Then, you can correct the errors and compile again.

Figure 2-9 How to compile a program and fix the compile-time errors

How to test a program

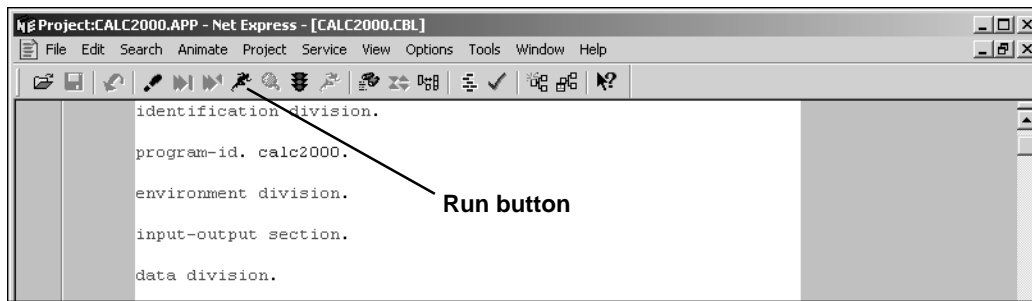
To *test* a program, you click on the Run button in the toolbar. Then, if the program uses Accept or Display statements, an *application output window* like the one in figure 2-10 is displayed. This is where the output of each Display statement is displayed. And this is where you enter the data for each Accept statement. For each entry, you just type the data and press the Enter key.

As you work with the application output window, remember that the program is running under control of Net Express. This can be referred to as *animating* a program. While your program is being animated, Net Express uses the application output window. In contrast, a production program is run from an executable so its output is displayed in a Windows application window.

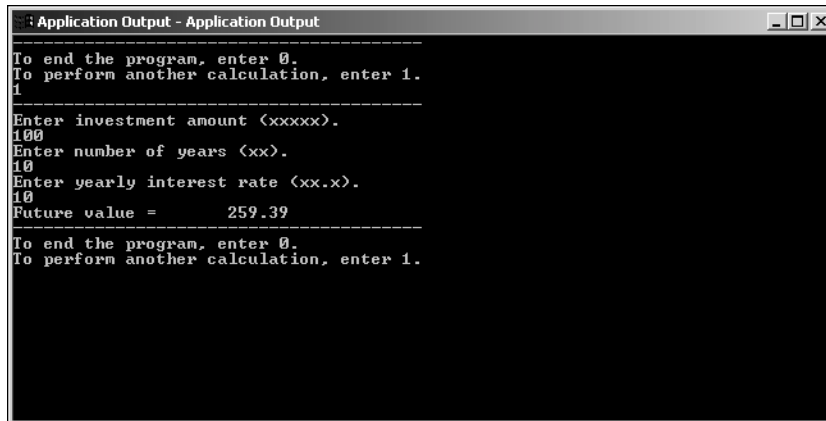
When a program runs until the Stop Run statement is executed, it is a *normal program termination*. But that doesn't mean the program is correct. You still have to study the output of the program to make sure that it worked correctly. If it didn't, you need to *debug* the program. As you debug a program, you may need to run it two or more times to determine why it isn't working correctly.

This figure also shows how to cancel the execution of a program. This is useful for canceling a program that's "caught in a loop." That can happen if a Perform Until statement is executing and the condition in the Until clause is never met. Unless you cancel out of the program, it will run indefinitely. To cancel the program, you use the Ctrl+Break key combination to stop the program and the Animate→Stop Animating command to stop animating the program.

The document window for the program to be run



The window that's displayed when you run an interactive program



How to start the execution of a program

- Click on the Run button. Then, click on the Start button in the Animating dialog box.

How to cancel the execution of a program

- Press Ctrl+Break to stop the execution of the program. Then, use the Animate→Stop Animating command to stop the animation of the program.

Description

- The goal of *testing* is to find the errors (or *bugs*) in a program. The goal of *debugging* is to fix those bugs so the program works correctly.
- When you test a program, you should try all possible combinations of input data to make sure the program will work correctly under all conditions.
- When Net Express runs a program, it is called *animating* a program. For this purpose, Net Express creates a Debug subfolder that it uses for storing the other files that it needs for animating the program.

Figure 2-10 How to test a program

How to correct run-time errors

If an error occurs during a test run, a dialog box like the one in figure 2-11 is displayed. Then, when you click on the OK button in that box, the program is put into *break mode*, you are returned to the document window, and the statement that caused the error is highlighted. This type of error is known as a *run-time error*.

In the example in this figure, the Perform statement is highlighted. Because the message in the dialog box says that the problem is an illegal character in a numeric field, though, you can assume that the problem is with one of the variables in the Until clause of that statement. There, the values in the two variables are being compared numerically to see whether the first is greater than the second.

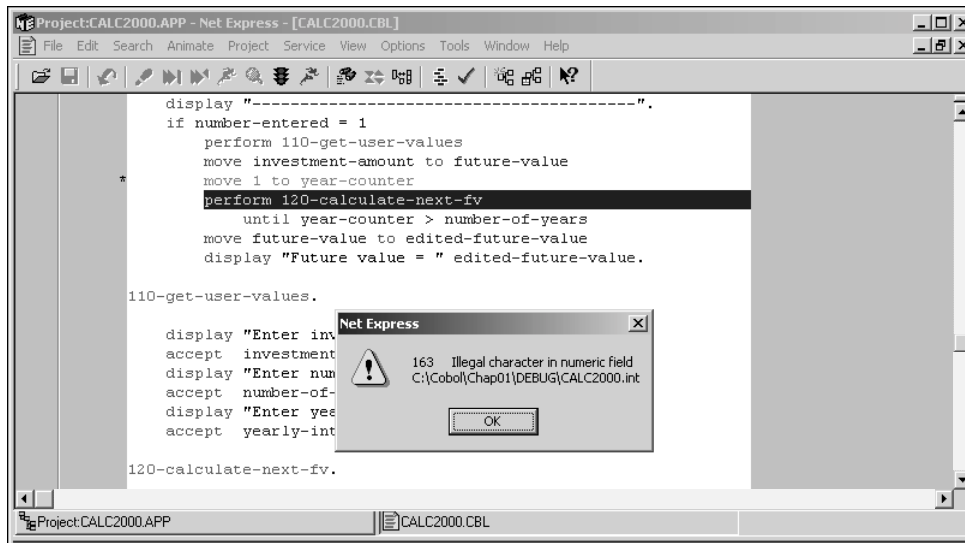
If you study the example, you can see that the statement before the highlighted one has an asterisk in column 7, so it's treated as a comment and ignored by the compiler. But this is the statement that sets the year-counter variable to 1. If this isn't done and a Value clause hasn't given it a starting value, the data will usually be invalid because it has the value of whatever was left in those storage positions by the last program.

Illegal data in a numeric field is the most common cause of run-time errors in COBOL. To help you figure out what caused the problem, you can display the current values of the variables while you're in break mode. You can also set breakpoints and step through a program to see exactly what's happening as the program executes. You'll learn those debugging skills in the next topics. When you figure out what the cause of the run-time error is, you can correct the error, recompile the program, and rerun it.

This figure also describes how Net Express handles two conditions that usually cause run-time errors with other compilers. The first condition occurs when the result of an arithmetic operation is too large for the receiving field. In that case, Net Express truncates the result. The second condition occurs when a Divide or Compute statement attempts to divide by zero. In that case, Net Express treats the result as zero.

Keep in mind that the results will be incorrect even though run-time errors won't occur for these conditions. That's why you need to check the results of all arithmetic operations to make sure that neither one of these conditions led to an error. In contrast, if these conditions do cause run-time errors as they do on most other compilers, you are forced to fix them. Either way, though, you need to check all results to make sure they're accurate.

The document window with the error statement highlighted



Description

- When a *run-time error* occurs, the program enters *break mode* and a dialog box that contains an error message is displayed.
- When you click on the OK button in the dialog box, you are returned to the document window with the statement that caused the error highlighted. Then, you can correct the error, recompile the program, and test it again.
- The most common cause of a run-time error is a statement that operates on invalid data.

How Net Express handles two other types of errors that normally cause run-time errors

- If an arithmetic operation has a result that is too large for the receiving field, the result is truncated instead of causing a run-time error.
- If a Divide or Compute statement tries to divide by zero, Net Express returns a result of zero instead of causing a run-time error.

Figure 2-11 How to correct run-time errors

How to use the debugging features

When you test a program and a run-time error occurs or the output isn't what you expect it to be, it can be difficult to locate the source of the errors just by looking at the code. That's why Net Express provides several debugging tools that can save you time and frustration as you test your programs. The topics that follow present the best of these tools.

How to display and modify the values of variables

When a program is in break mode, you can display the value of any variable by moving the mouse pointer over it. This displays a data tip that gives information like the picture and value of the variable.

Another way to display the value of any variable is to double-click on its name when the program is in break mode. This opens an Examine List dialog box like the one in figure 2-12. Here, you can see that the year-counter variable contained invalid numeric data when the program went into break mode. That, of course, is what caused the run-time error shown in the previous figure.

You can use the buttons on the right side of the Examine List dialog box to determine what else you do with the variable that's displayed. If, for example, you want to add the variable to the list at the bottom of the IDE, you can click on the Add to list button. This list will be displayed whenever the program enters break mode in future executions of the program.

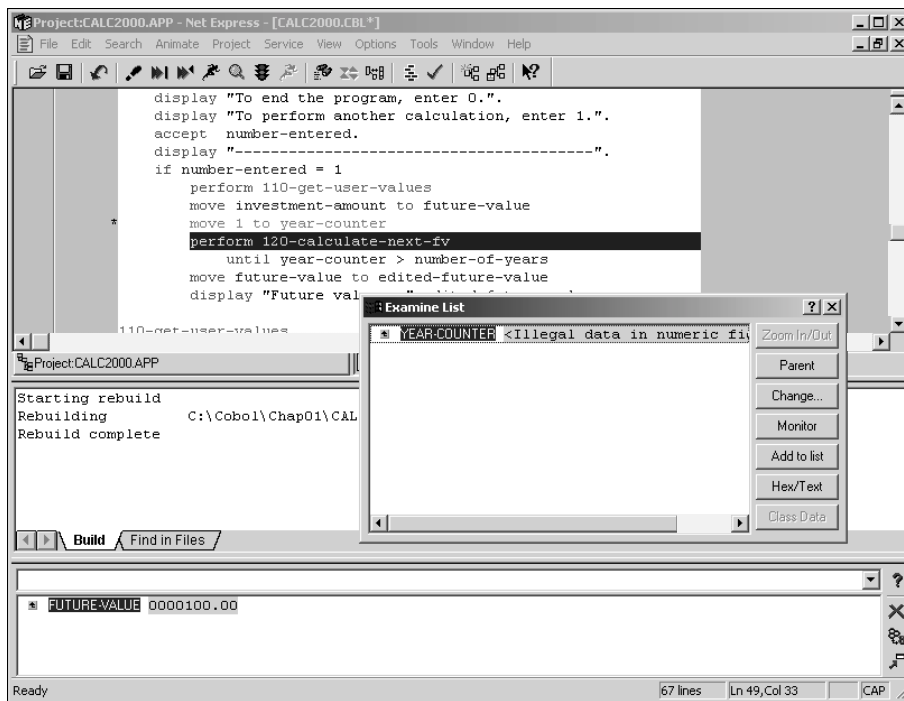
The Hex/Text button converts the value of a variable from text to *hexidecimal format* with two *hex digits* for each text character. If, for example, you click this button in the dialog box in this figure, the display changes to:

```
YEAR-COUNTER 20 20 20
```

Since hex 20 is the ASCII code for a blank, the value in this numeric field is three blanks, which is an invalid numeric value. To learn more about hex codes, please refer to chapter 6.

If you want to change the value of a variable while a program is in break mode, you can do that by clicking on the Change button in the Examine List dialog box. In some cases, after you change the value, you can click on the Run button to continue the test run with the new value.

The document window while the program is in break mode



Two ways to display the value of any variable while in break mode

- Move the mouse pointer over the variable name. This displays a data tip that gives the current value of that variable.
- Double-click on any variable name in the source code. This opens the Examine List dialog box, which gives the current value of that variable.

How to use the buttons in the Examine List dialog box

- To add a variable to the list at the bottom of the IDE, click on the Add to list button. The variable values in this list will be changed as the program runs so you can monitor them.
- To open a small dialog box that monitors the value for a single variable, click on the Monitor button.
- To change the value of a variable, click on the Change button. Then, type the new value in the dialog box that's displayed and click on the Apply button.
- To convert data from text to *hexadecimal format*, click on the Hex/Text button. In hex format, each pair of *hex digits* represents the data for one byte of storage.

Figure 2-12 How to display and modify the values of variables

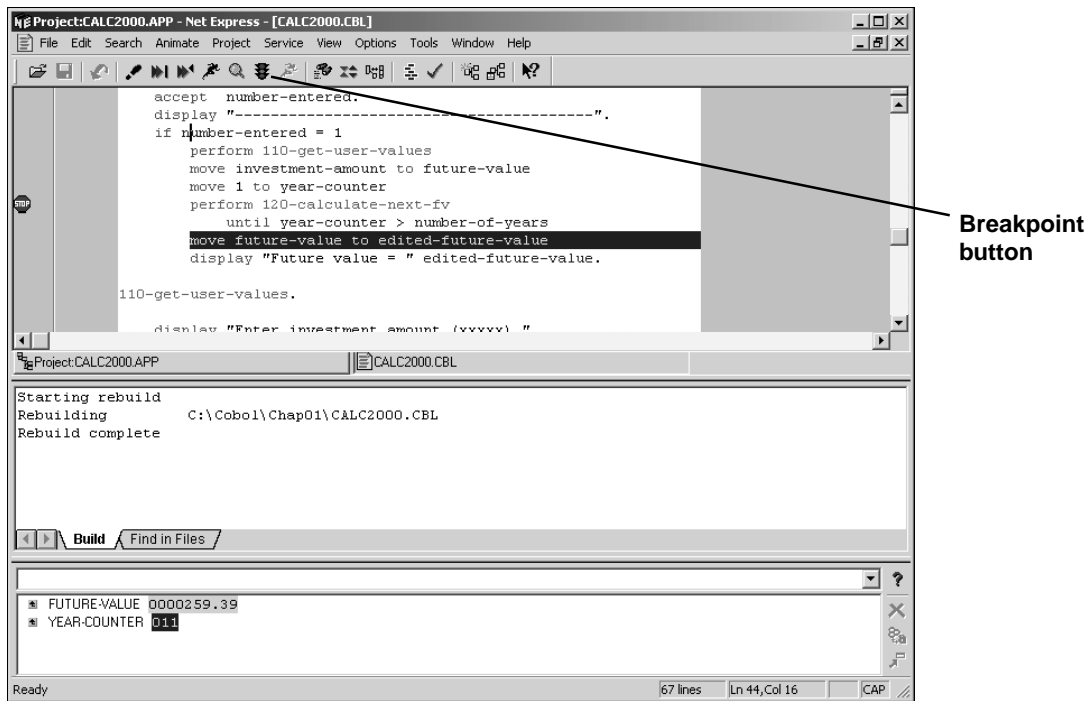
How to use breakpoints

When you set a *breakpoint* on a statement as shown in figure 2-13, the program enters break mode each time it comes to that statement. In this figure, for example, a breakpoint has been set on the Perform Until statement. This is indicated by the stop sign in the left margin.

While in break mode, you can display and change the values of variables. In this figure, the future-value and year-counter variables have been added to the list at the bottom of the IDE. Then, you can analyze the way this data changes to make sure that it's working correctly. If necessary, you can set two or more breakpoints in a program so you can monitor the data at all of the program's critical points.

When you use breakpoints or step through a program as described in the next figure, Net Express displays a dialog box when the Stop Run statement is executed. Then, before you can rerun the program by clicking on the Run button, you need to close the dialog box and execute the Animate→Stop Animating command.

The document window when a breakpoint is reached



How to set and remove breakpoints

- To set or remove a breakpoint, you can double-click in the margin to the left of a statement or select the statement and click the Breakpoint toolbar button. To remove all breakpoints, choose the Animate→Breakpoint→Clear All In Program command.

Description

- You can set a *breakpoint* on any executable COBOL statement or paragraph heading in the Procedure Division. Program execution stops when it reaches the statement you marked or the first statement in the paragraph you marked.
- When a program reaches a breakpoint, the program enters break mode. Then, you can display, monitor, and change the values of variables. You can also set and remove breakpoints. When you're ready to continue, you can click on the Run button. Or, you can step through the instructions after the breakpoint as shown in the next figure.
- When you use breakpoints and the program reaches the Stop Run statement, a dialog box is displayed. Then, you can click on the OK button to close that box. Before you can run the program again, you need to run the Animate→Stop Animating command.

Figure 2-13 How to use breakpoints

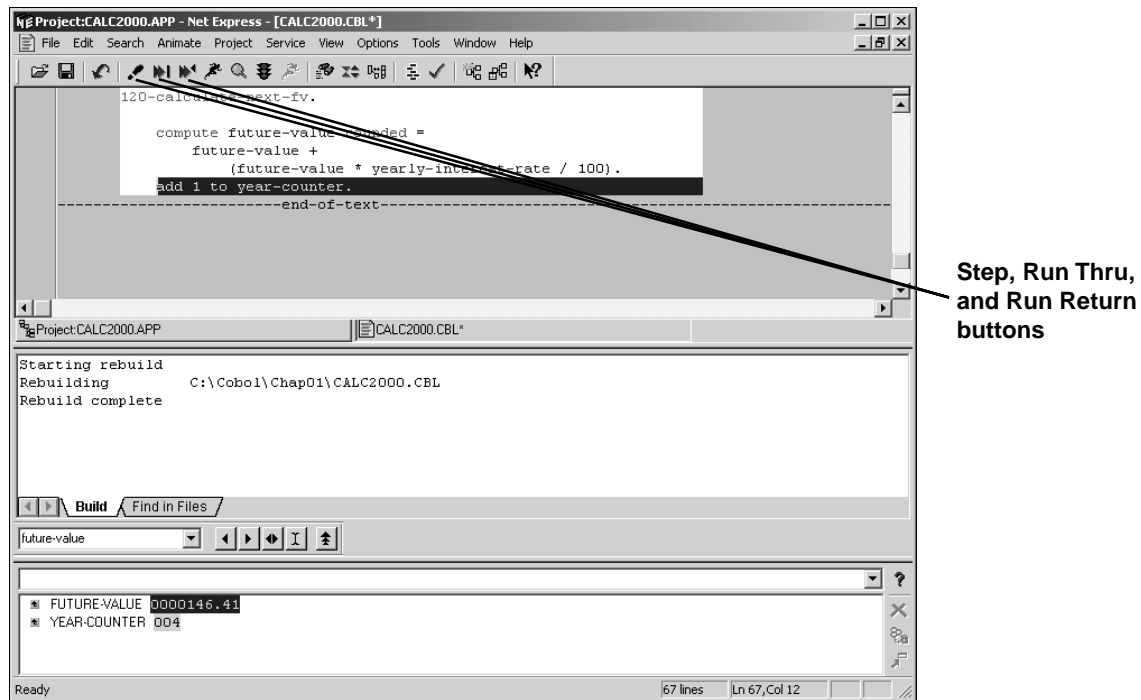
How to step through a program

From break mode or from the start of a program, you can *step through the program* instead of running all the statements without intervention. Then, you can see how the values of the variables change as each statement or group of statements is executed.

Figure 2-14 identifies the three buttons that you can use as you step through a program. The Step button lets you run one statement at a time and re-enters break mode at the end of each statement. The Run Thru button lets you run through all of the statements in a performed procedure before re-entering break mode. And the Run Return statement lets you execute the remaining statements in a performed procedure before re-entering break mode.

The breakpoint and step features in combination with the features for displaying the values of variables can save you many hours of debugging time. In addition, these features are wonderful learning tools. By stepping through code that you don't understand and displaying the values of related variables, you can see exactly how a program works.

The next statement is highlighted when you step through a program



How to step through a program

- To execute just the highlighted statement, click on the Step button. After the statement is executed, the program enters break mode again at the next statement.
- To execute the statements in a performed procedure without stopping, click on the Run Thru button. After the performed statements are executed, the program enters break mode at the next statement after the Perform statement.
- To execute the rest of the statements in the called procedure without stopping, click on the Run Return button. Then, the program enters break mode at the statement after the Perform statement that called the procedure.
- To execute all the remaining statements in the program without stopping, remove all the breakpoints and click on the Run button.

Figure 2-14 How to step through a program

Perspective

Now that you know how to compile and test a program using Micro Focus Net Express, you should be able to write simple interactive programs of your own. Then, in the next chapter, you will learn how to write programs that read files and prepare reports.

As you work with Net Express, you should know that many mainframe programmers use Micro Focus Mainframe Express to develop COBOL programs on their PCs. After they compile and test their programs with Mainframe Express, they use its tools to upload their programs and data to the mainframe for final testing. Because Mainframe Express provides a friendlier environment and better debugging tools than a mainframe, developing programs in this way can be much more efficient than developing them directly on the mainframe.

Summary

- You can use the University Edition of Net Express to develop COBOL programs on a PC. The IDE for Net Express is similar to the IDE for Micro Focus Mainframe Express, which can be used for developing mainframe programs on a PC.
- When you use Net Express, each *project* consists of one or more COBOL *source files*. When you open the project, all of the files for the project are displayed in the *project window*.
- To enter, edit, and *compile* the source code for a program, you use the *document window*. When the program compiles without errors, called a *clean compile*, you can *test* and *debug* the program.
- A *compile-time error* occurs when the compiler can't compile a statement because its syntax is incorrect. A *run-time error* occurs when Net Express is unable to execute a statement.
- To help you debug a program, Net Express provides debugging tools. These tools let you set *breakpoints*, display and change the values of variables while the program is in *break mode*, and *step through a program*.

Terms

Integrated Development Environment (IDE)	COBOL profile
project window	tagged line
project	compress
document window	expand
output window	compile-time error
source code	clean compile
source file	test
intermediate code	application output window
intermediate file	animate
compile	normal program termination
execute	debug
run	bug
production program	run-time error
executable	break mode
COBOL compiler	hexadecimal format
object module	hex digit
diagnostics	breakpoint
linkage editor	step through a program
link edit (or link)	

Objectives

- Given the specifications for a simple interactive program like the ones in chapter 1, use Micro Focus Net Express to enter, compile, test, and debug the program.
- Describe the difference between testing and debugging.
- Describe the difference between compile-time and run-time errors.
- Explain why you should start your new programs from old programs.
- Describe the use of these debugging features: breakpoints, displaying variable values in break mode, and stepping through a program.