

How to use Net Express 4.0 to develop COBOL programs

This document shows you how to develop COBOL programs using the University Edition of Net Express 4.0. Net Express is a relatively inexpensive tool that you can use to learn COBOL on your own PC. So even if you're using a different compiler on the job or in your school lab, you may want to get a copy of Net Express so you can practice on your own PC.

This document is designed as a supplement to our book, *Murach's Mainframe COBOL*, which teaches you everything you need to know to develop COBOL programs on an IBM mainframe. This document is divided into two units. In unit 1, you'll learn how to use Net Express to edit, compile, test, and debug COBOL programs. And in unit 2, you'll learn how to use Net Express to develop COBOL programs that work with files. When you complete this tutorial, you'll be able to develop programs on your own PC using that compiler.



Copyright © 2004 Mike Murach & Associates, Inc.
All rights reserved.

murachbooks@murach.com • www.murach.com

Unit 1

How to edit, compile, test, and debug programs

In this unit, you'll learn how to edit, compile, test, and debug a program using Net Express 4.0. It is designed to replace chapter 2 of *Murach's Mainframe COBOL*. That chapter presents the terms and concepts that prepare you for learning how to develop COBOL programs on a specific computer and compiler. If you'll be using Net Express 4.0 on a PC, then, you'll want to read this unit instead.

Introduction to the Net Express 4.0 IDE	4
The Integrated Development Environment of Net Express 4.0	4
How a production program is compiled, link edited, and executed	6
How to enter and edit a program	8
How to open an existing project	8
How the project files and folders can be organized	8
How to start a new project	10
How to add files to a project	12
How to set the default folder	14
How to enter and edit source code	16
How to find and replace text	18
How to print the source code for a program	18
How to compile and test a program	20
How to compile a program	20
How to correct compile-time errors	20
How to test a program	22
How to correct run-time errors	24
How to use the debugging features	26
How to display and modify the values of variables	26
How to use breakpoints	28
How to step through a program	30
Perspective	32

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.

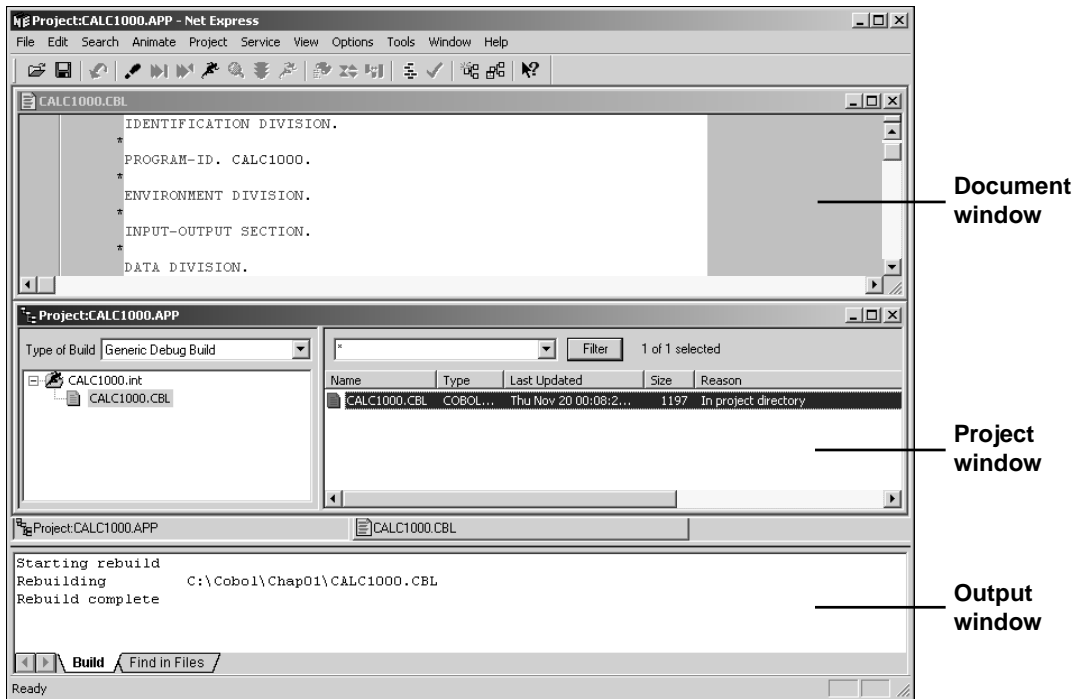
The Integrated Development Environment of Net Express 4.0

When you develop programs with the IDE for Net Express 4.0, you work with the three main windows shown in figure 1-1. 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 our 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 document, 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 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. In many cases, 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 Rebuild 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 1-1 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 1-2 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.

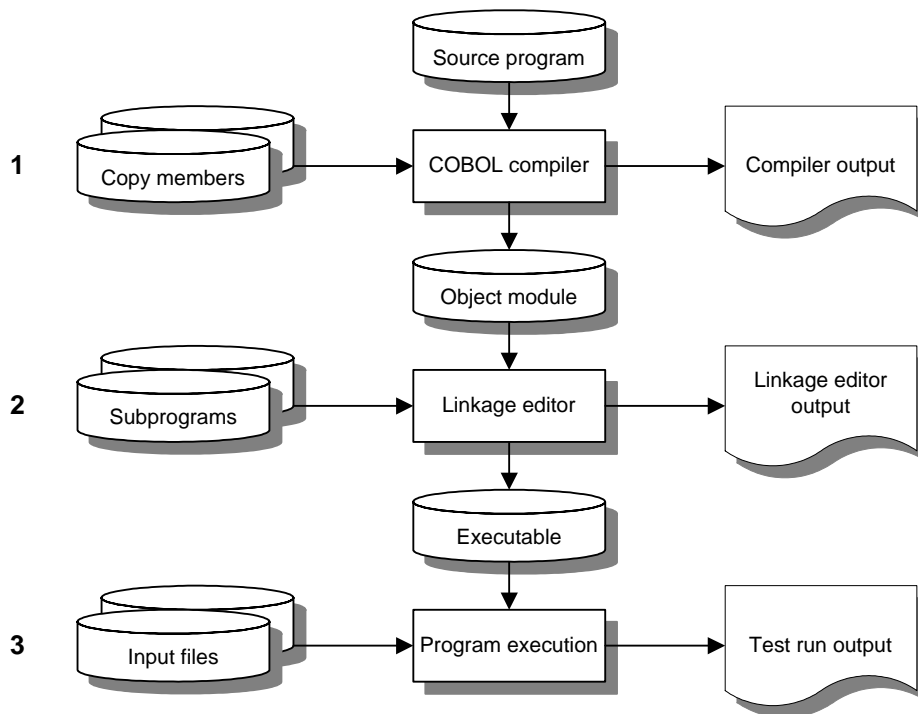
In step 1 in this figure, the COBOL compiler compiles the source program into an *object module*. If the source program uses any Copy statements (see chapter 11 of our book), the compiler copies the source code from the related copy members into the program as part of this process. The compiler also 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 of our book). 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 of our book.

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 of our book). 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 of our book). 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 1-2 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 1-3 to open the project. First, when you start Net Express, the last project you were working on is opened automatically. If you want to work on another project, 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, the COBOL files in that project are displayed in the project window as you saw in figure 1-1. Then, you can open the document window for a file by double-clicking on it.

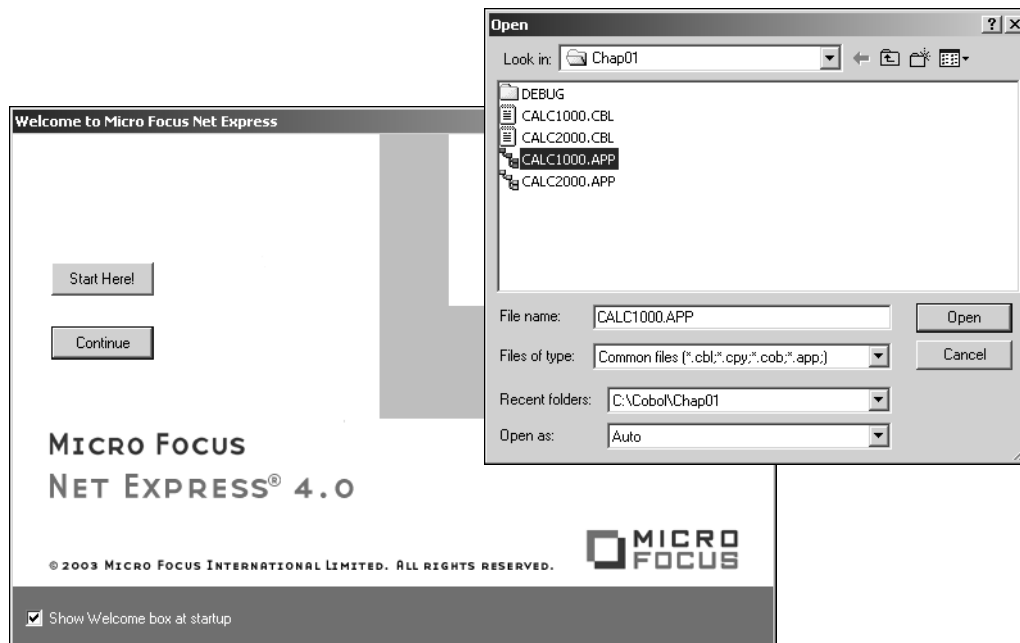
How the project files and folders can be organized

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 1-3 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

- Start Net Express, and the project that you were working on when you ended your last Net Express session will be opened. If the Welcome dialog box is displayed, click on the Continue button to close it.
- 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.

Note

- If you don't want the Welcome dialog box to be displayed each time you start Net Express, you can remove the check mark from the Show Welcome box at startup option in this dialog box.

Figure 1-3 How to open an existing project

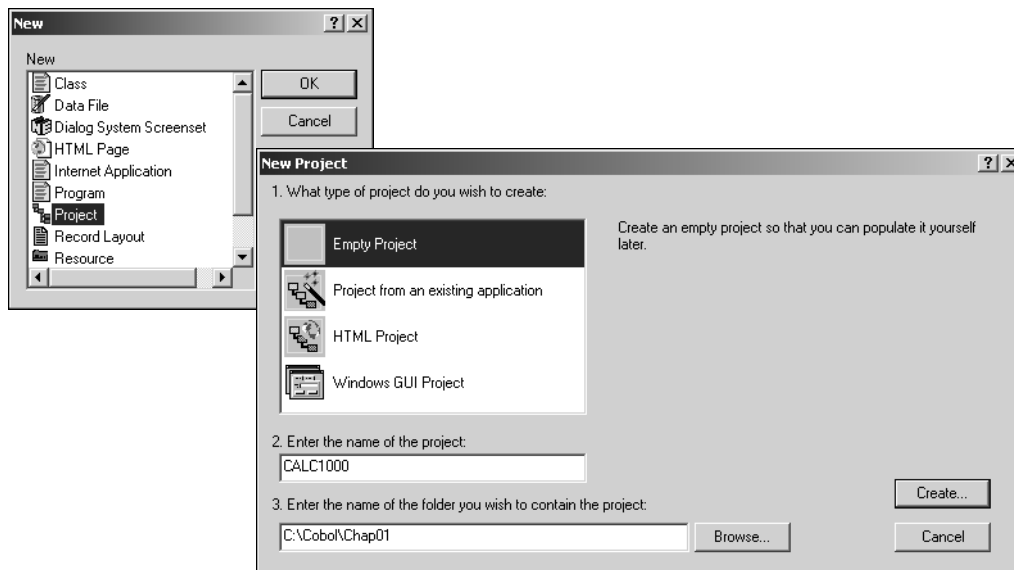
How to start a new project

Before you start a new program, you need to start a new project. To do that, you use the procedure in figure 1-4. 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 1-4 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 1-5.

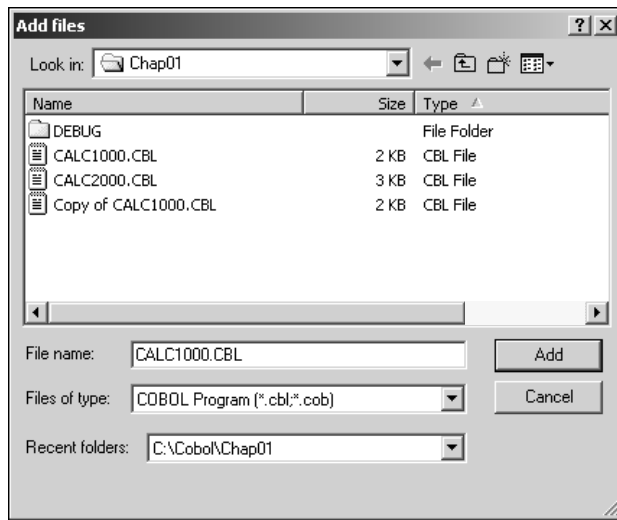
In the Add files dialog box, if you specify a file name that doesn't already exist, Net Express will create an empty file with that name. Then, you can enter COBOL source code into that file. Because every COBOL program requires some of the same coding, though, you should rarely, if ever, start a new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

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

1. Use the controls in the Add files dialog box to move to the folder that contains the file that you want to copy. Then, right-click on the file and select the Copy command. This copies the file to the clipboard.
2. Use the controls in the Add files 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 to 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 1-5 How add files to a project

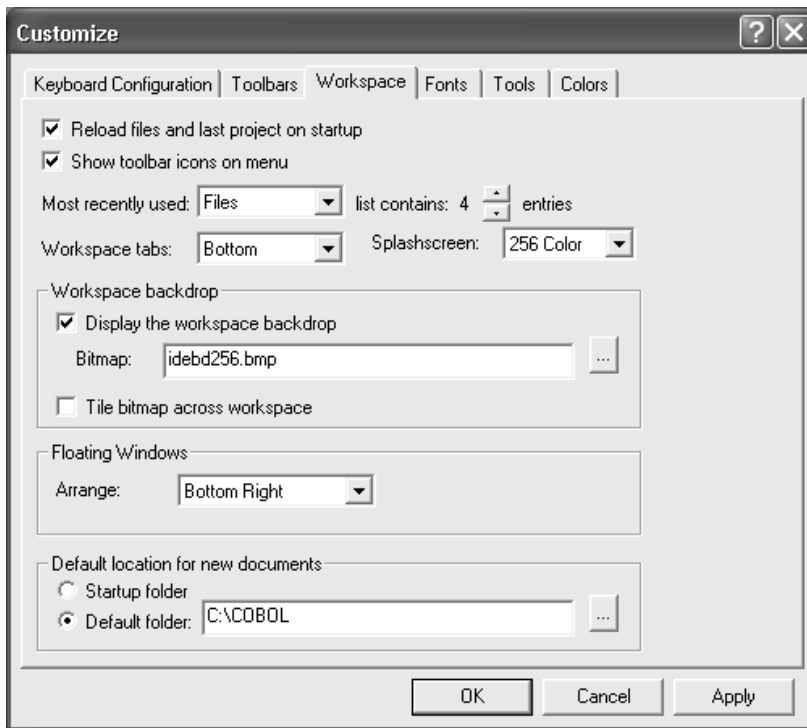
How to set the default folder

To make it easier to create new files, 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 1-6, you use the Workspace tab of the Customize dialog box to change the default folder. In this dialog box, you enter the path for the folder that you want to use as the default in the Default folder box. In this figure, the default folder is being changed to `C:\COBOL`. Note that since the path isn't case sensitive, you don't have to worry about the capitalization.

After you set the default folder, any file you create is stored in that folder unless you explicitly specify another folder. Note, however, that the default folder isn't used when you work with existing files. Instead, the default folder for existing files is the one that was used most recently. If the last project you opened was in the `C:\Cobol\Chap01` folder, for example, that's the folder that's displayed by default the next time you open a project or add a new file to a project. That's true even after you close and restart Net Express.

The Customize dialog box for the Net Express IDE



How to change the default folder

- Use the Options→Customize IDE command to display the Customize dialog box. Then, click on the Workspace tab and enter the folder you want to use as the default location for new documents in the Default folder option box.

Description

- When you install Net Express, the default folder is set to your My Documents folder. For efficiency, though, you should change the default to the top folder for the projects that you're developing.

Note

- The default folder isn't used when working with existing files. Instead, the default folder is the last folder that a file was retrieved from.

Figure 1-6 How to set the default folder

How to enter and edit source code

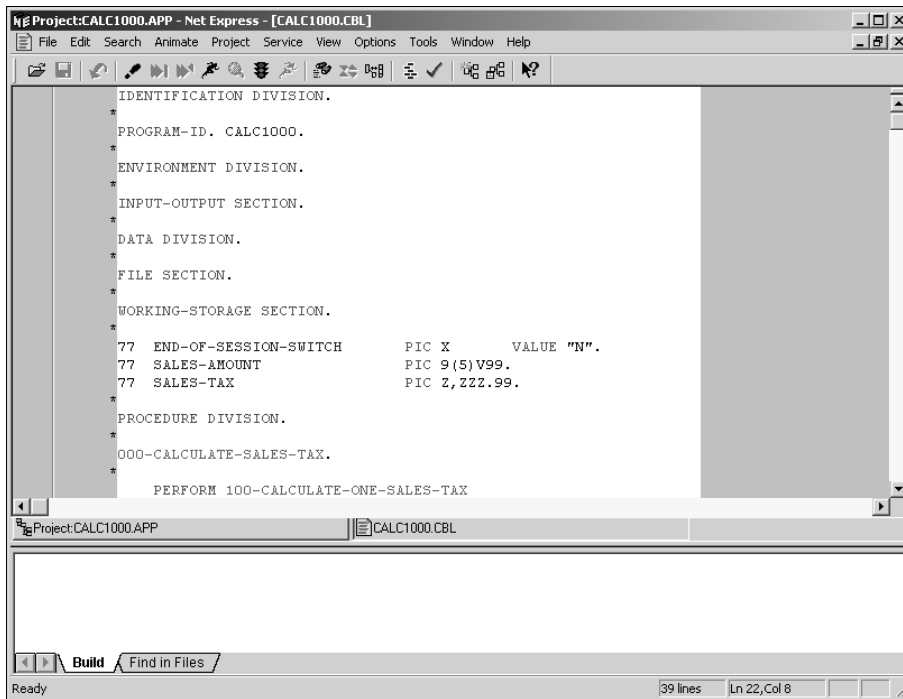
Once you've created a new project file and new program file, you can double-click on the COBOL file in the project window to open the document window for the file. This window is shown in figure 1-7. Notice that the left and right margins are set the way you want them for a COBOL program because that's 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.

To move from one window to another, you can use normal Windows techniques like pressing Ctrl+F6 or using the commands in the Window menu. You can also click on the tabs at the bottom of the main window (the one that contains the project and document windows) 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 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 1-7 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 1-8. 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.

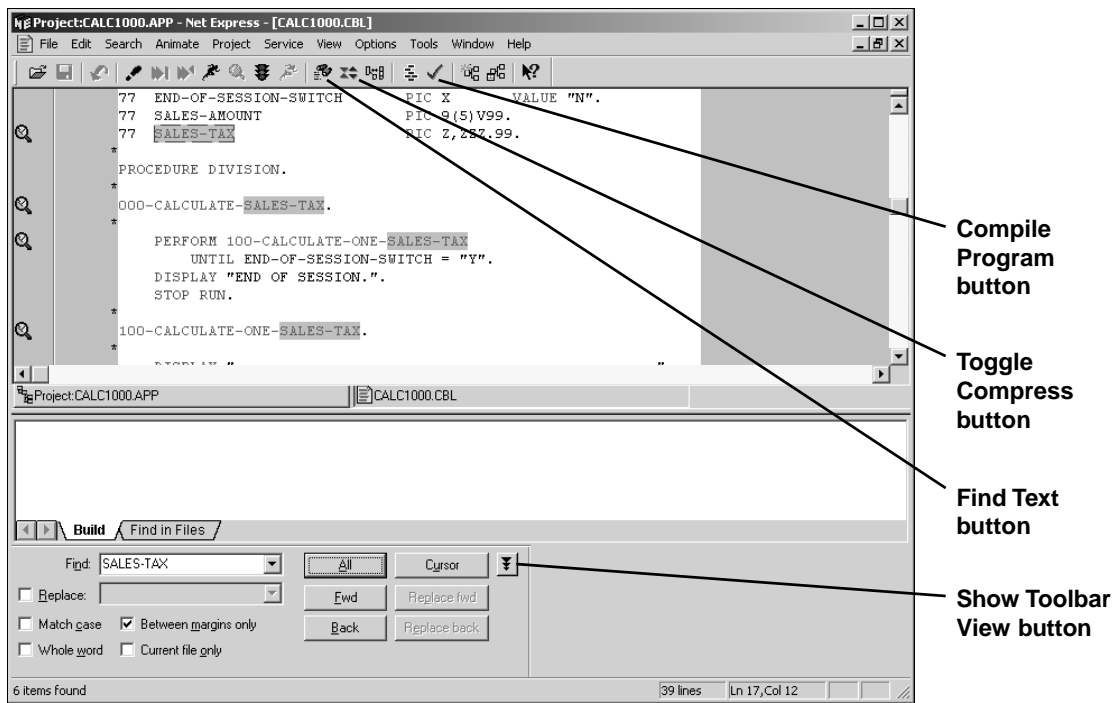
When you click on the All button to find all the occurrences of the text, the lines that contain those occurrences are tagged and highlighted as shown in this figure. Then, you can move from one *tagged line* to another. You can also *compress* the source code so only the tagged lines appear in the document window, after which you can *expand* the source code 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 Find Text command has been executed



How to use the Find Text 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 button. 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 1-8 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 1-9 shows how to compile a program. The easiest way to start the compile is to click on the Rebuild 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.

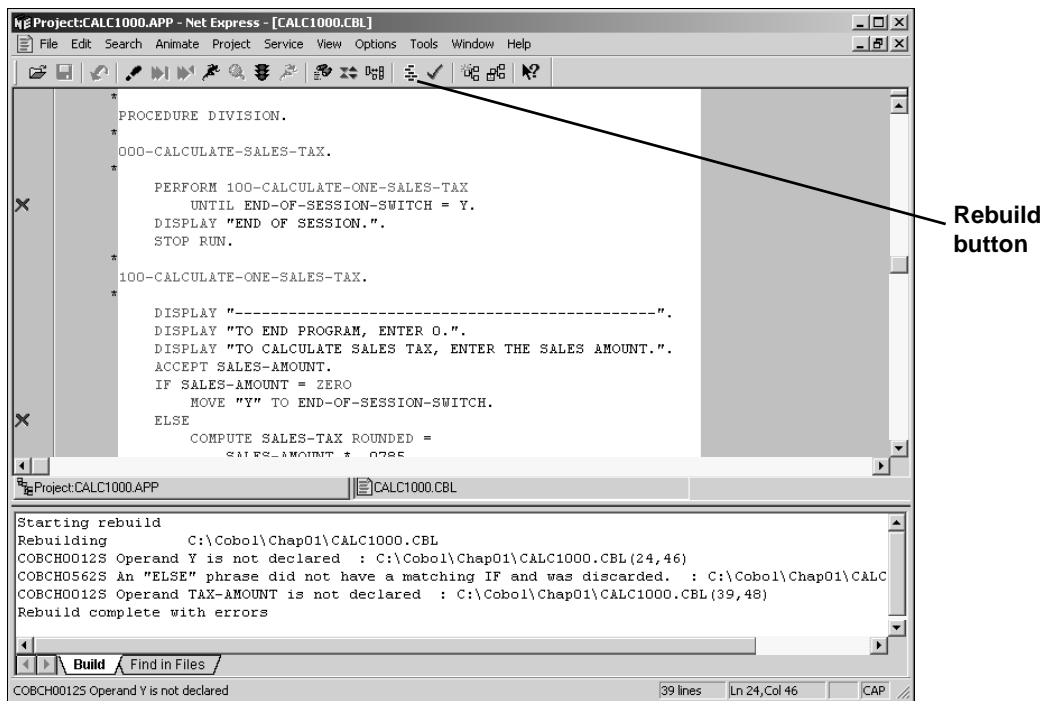
How to correct compile-time errors

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.

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 Rebuild 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 output 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 1-9 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. When you do, the Start Animating dialog box is displayed by default. This dialog box lets you select the program you want to run, and it lets you change options that will be used during program execution. In most cases, you'll just click the OK button in this dialog box to run the current program with the default options.

If the program uses Accept or Display statements, an *application output window* like the one in figure 1-10 is displayed on top of the Net Express window. 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.

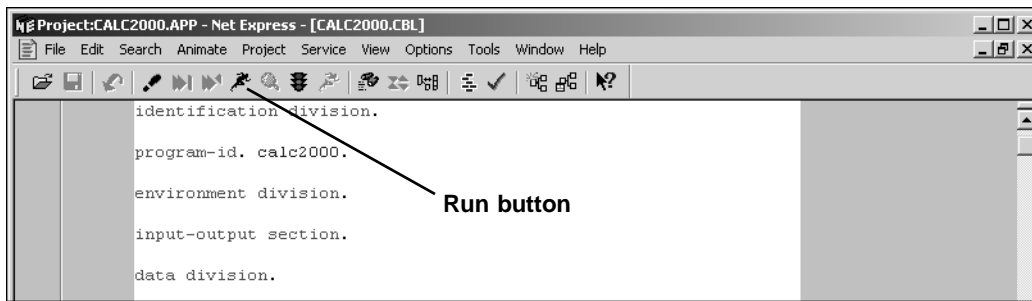
As you work with the application output window, remember that the program is running under control of Net Express. This can be referred to as *animating* a program. While your program is being animated, Net Express uses the application output window. In contrast, a production program is run from an executable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

When a program runs until the Stop Run statement is executed, animation ends automatically. This is called 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.

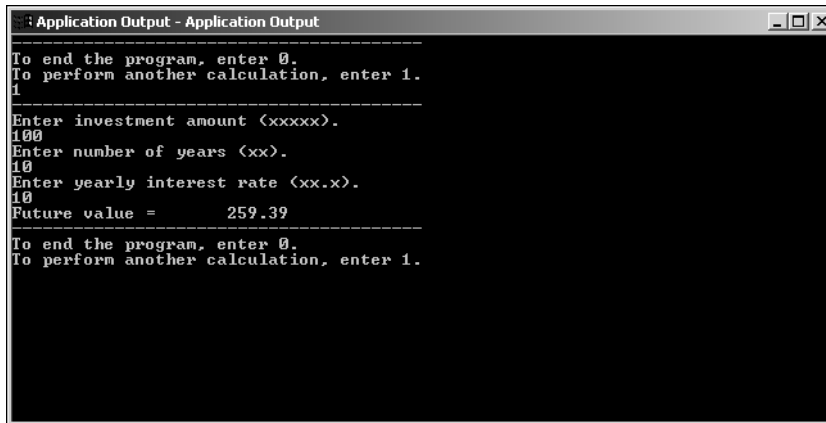
This figure also shows how to cancel the execution of a program. This is useful for canceling a program that's "caught in a loop." That can happen if a Perform Until statement is executing and the condition in the Until clause is never met. Unless you cancel out of the program, it will run indefinitely. 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.

Whether a program terminates normally or you terminate it using the Animate→Stop Animating command, the application output window remains open. If you want to close this window, you can click on the Close button in its upper right corner. Or, you can right-click in the window to display its shortcut menu and then select the Hide command. In most cases, though, you'll just leave the window open and switch back to Net Express to continue working.

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 OK button in the Start 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.

How to correct run-time errors

If an error occurs during a test run, a dialog box like the one in figure 1-11 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.

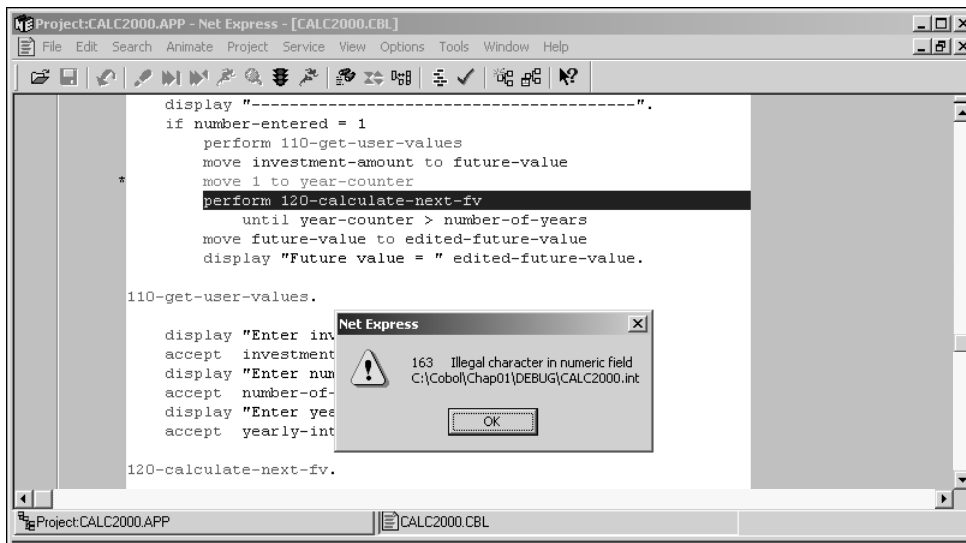
If you study the example, you can see that the statement before the highlighted one has an asterisk in column 7, so it's treated as a comment and ignored by the compiler. But this is the statement that sets the year-counter variable to 1. If this isn't done and a Value clause hasn't given it a starting 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 1-11 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 1-12. 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.

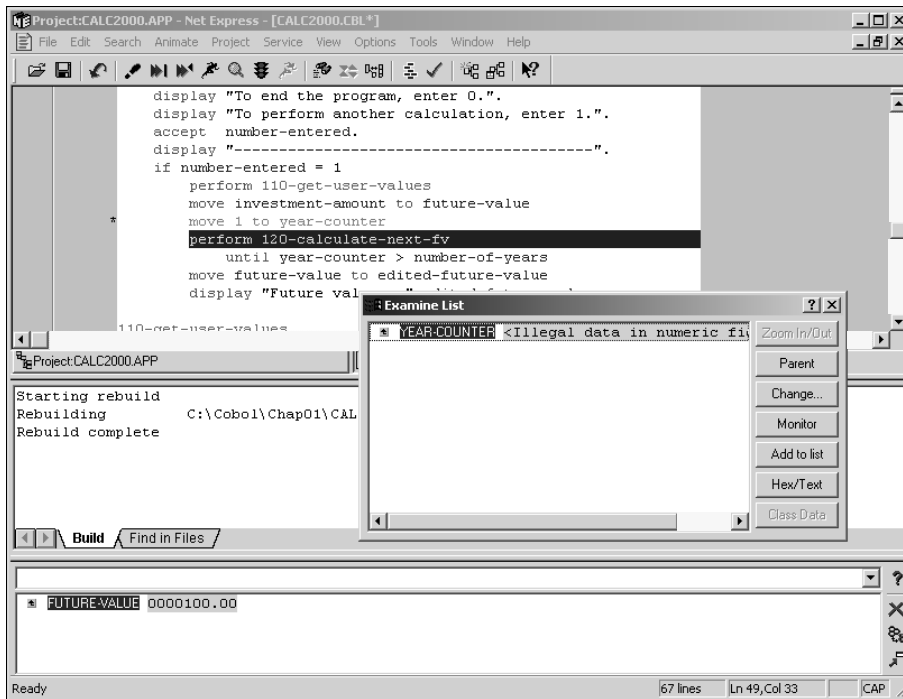
The Hex/Text button converts the value of a variable from text to *hexidecimal format* with two *hex digits* for each text character. If, for example, you click this button in the dialog box in this figure, the display changes to:

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 of our book.

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 updated 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 1-12 How to display and modify the values of variables

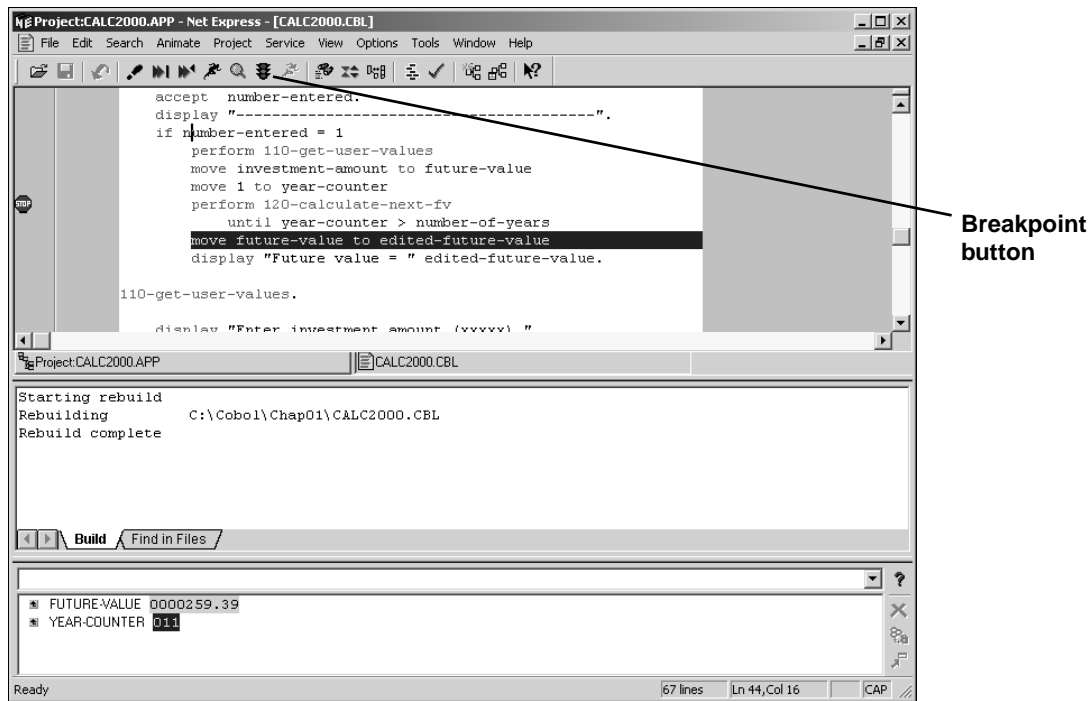
How to use breakpoints

When you set a *breakpoint* on a statement as shown in figure 1-13, 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 1-13 How to use breakpoints

How to step through a program

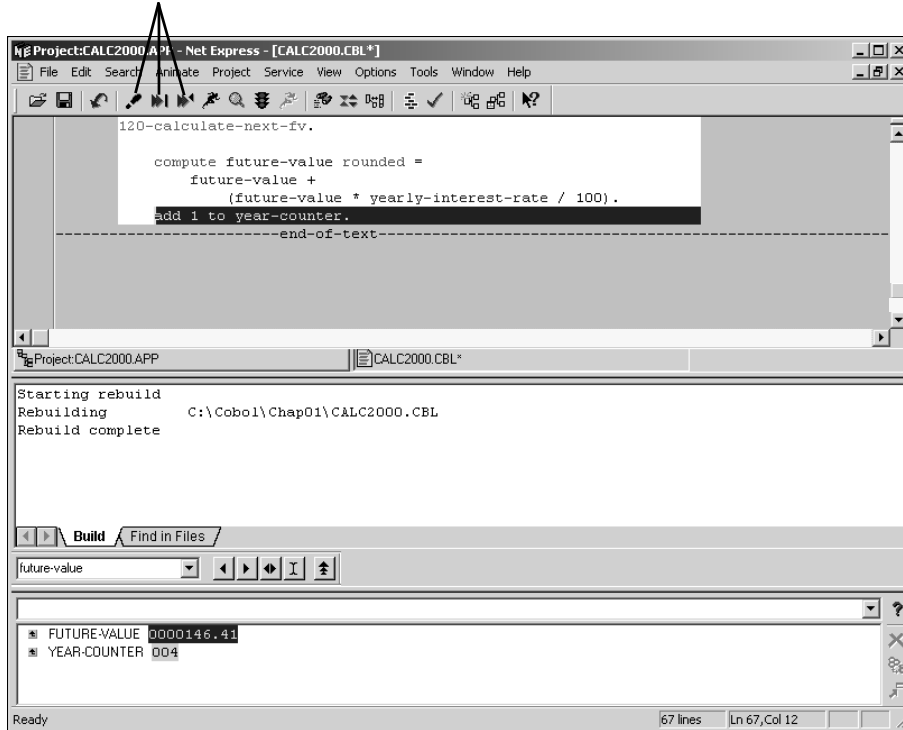
From break mode or from the start of a program, you can *step through the program* instead of running all the statements without intervention. Then, you can see how the values of the variables change as each statement or group of statements is executed.

Figure 1-14 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

Step, Run Thru, and
Run Return buttons



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 a performed procedure without stopping, click on the Run Return button. Then, the program enters break mode at the statement after the Perform statement that performed the procedure.
- To execute all the remaining statements in the program without stopping, remove all the breakpoints and click on the Run button.

Figure 1-14 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 write simple interactive programs of your own. Then, if you haven't already done so, you can read the chapters in the first three sections of *Murach's Mainframe COBOL*. These chapters will teach you all the skills you need to develop professional COBOL programs.

As you work with Net Express, you should know that Micro Focus has a related product called Micro Focus Mainframe Express that's designed to help mainframe programmers develop COBOL programs on their PCs. After compiling and testing a program with Mainframe Express, a programmer can use its tools to upload the program 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 the source code
document window	expand the source code
output window	compile-time error
source code	clean compile
source file	test
intermediate code	application output window
intermediate file	animate a program
compile	normal program termination
execute a program	debug
run a program	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 of our book, 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.

Unit 2

How to develop programs that work with files

In this unit, you'll learn the basic skills for using Net Express to develop programs that work with files. That includes coding Select statements that refer to files, displaying and working with data files and print files, creating test files, and using copy members. If you compare this information to the corresponding information in *Murach's Mainframe COBOL*, you'll see that it's significantly easier to work with files on a PC than on a mainframe. That's why we recommend you use Net Express to learn COBOL programming.

As you read this unit, keep in mind that it doesn't present all the features Net Express provides for working with files. Instead, it presents just those that we think you'll find most useful. To find out what other features are available, you can refer to the help documentation that comes with Net Express.

How to code Select statements	36
How to code Select statements for sequential files	36
How to code Select statements for indexed files	38
How to code Select statements for relative files	38
How to code Select statements for sort work files	38
How to display and work with the data in a file	40
How to display and work with the data in a sequential file	40
How to display and work with the data in an indexed or relative file	42
How to display and work with the data in a print file	44
How to create test files	46
How to create a sequential file	46
How to create an indexed file	48
How to create a relative file	50
How to use copy members	52
How to create a copy member	52
How to code the Copy statement	54
Perspective	56

How to code Select statements

For each file that a program reads or writes, you need to code one Select statement in the Environment Division. You code these statements differently when you use Net Express than you do when you use mainframe COBOL. In the topics that follow, you'll see examples of Select statements for all the different types of files you can use with Net Express.

How to code Select statements for sequential files

Net Express provides for the three types of sequential files shown in figure 2-1. The type of file you're most likely to use is a *record sequential file*, also called just a *sequential file*. This type of file consists of a single string of characters. In other words, there's no indication of where each field or record ends. This information is provided by the record description you include in your program. Notice that when you code a Select statement for a record sequential file, you can omit the Organization clause.

The second type of sequential file is the *line sequential file*. To use this type of file, you must include the Organization is Line Sequential clause as illustrated by the example.

Unlike record sequential files, the records in a line sequential file end with a paragraph mark (a carriage return plus a line feed). Because of that, they're easier to read than sequential files when you display them in a text editor or word processing program. As you'll see later in this chapter, though, Net Express provides a feature that makes it easy to display and read any type of file. So you'll typically use a line sequential file only if you want to display the file from outside of Net Express.

The third type of sequential file is the *printer sequential file*. Actually, a printer sequential file, also called a *printer file* or *print file*, is a special kind of record sequential file. In fact, if you omit the Line Advancing File clause, the Select statement for a print file looks just like the Select statement for a record sequential file.

Each record in a print file contains the information to be printed on one line and ends with a line feed. Between the records are characters that indicate the vertical position of the record that follows. For example, one character might indicate that a line should be skipped before the next line is printed, and another might indicate that the next line should be printed at the top of the next page.

When you code a Select statement for a sequential file, you must include a *system name* that identifies the file on disk. As you can see in this figure, the system name includes the path and the file name for the file. Notice that the file name for a data file is typically given an extension of *dat*, and a file name for a print file is typically given an extension of *prn*. That makes it easy to tell them apart.

How to code the Select statement for a record sequential file

The syntax of the Select statement for a record sequential file

```
SELECT file-name ASSIGN TO system-name
      [ORGANIZATION IS RECORD SEQUENTIAL]
```

A typical Select statement for a record sequential file

```
SELECT CUSTMAST ASSIGN TO "c:\cobol\data\custmast.dat".
```

How to code the Select statement for a line sequential file

The syntax of the Select statement for a line sequential file

```
SELECT file-name ASSIGN TO system-name
      ORGANIZATION IS LINE SEQUENTIAL
```

A typical Select statement for a line sequential file

```
SELECT MNTTRAN ASSIGN TO "c:\cobol\data\mnttran.dat"
      ORGANIZATION IS LINE SEQUENTIAL.
```

How to code the Select statement for a printer sequential file

The syntax of the Select statement for a printer sequential file

```
SELECT file-name ASSIGN TO [LINE ADVANCING FILE] system-name
```

A typical Select statement for a printer sequential file

```
SELECT SALESRPT ASSIGN TO "c:\cobol\data\salesrpt.prn".
```

The syntax for a system name

```
"path\filename"
```

Description

- A Select statement identifies a disk file or a print file used by the program.
- The *system name* for a sequential file consists of the path and file name for the file.
- Net Express provides for three types of sequential files: record sequential files, line sequential files, and printer sequential files.
- A *record sequential file*, also called just a *sequential file*, consists of a single string of characters.
- In a *line sequential file*, each record ends with a record delimiter that consists of a carriage return and line feed.
- A *printer sequential file*, or just *printer* or *print file*, is a special type of record sequential file that consists of print records. Each record ends with a carriage return, and the records are separated by vertical positioning characters.
- The file name for a record or line sequential file is typically given an extension of *dat*, and the file name for a printer sequential file is typically given an extension of *prn*.
- Although you can also code the system name for a print file so the output goes directly to a printer, it's better to code the system name so it is saved in a disk file. Then, you can use your word processor or NotePad to print the output or to review the output on your monitor without ever printing it (see figure 2-5).

Figure 2-1 How to code Select statements for sequential files

How to code Select statements for indexed files

At the top of figure 2-2, you can see three examples of Select statements for indexed files. Because the syntax of a Select statement for an indexed file is the same for Net Express as it is for a mainframe, it isn't presented here. The only difference is the way you code the system name. Like a sequential file, the system name for an indexed file consists of a path and a file name.

The first two examples in this figure illustrate two ways that you can access an indexed file. As you can tell from the Access clause, the first statement provides for sequential access, and the second statement provides for random access. You can also access an indexed file both sequentially and randomly in the same program by specifying the DYNAMIC keyword on the Access clause. Regardless of the access you use, you must identify the primary key using the Record Key clause.

The third example shows how you code a Select statement for a file that contains an alternate index. To do that, you include an Alternate Record Key clause. Note that you must code this clause whether or not the program uses the alternate index. That's different from a mainframe, in which case you have to code this clause only if you want to access records using the alternate index or the program updates records and you want the alternate index updated.

How to code Select statements for relative files

Figure 2-2 also presents two examples of Select statements for relative files. Like an indexed file, the only difference in the way you code a Select statement for a relative file in a Net Express program versus a mainframe program is how you code the system name.

The first example in this figure shows how to access a relative file sequentially. Notice that when you do that, you don't have to name the field that contains the relative record number. When you access a relative file randomly, however, you have to name that field in the Relative Key clause as shown in the second example. Although it's not common, you can also access a relative file both sequentially and randomly in the same program using dynamic access.

How to code Select statements for sort work files

The last Select statement in figure 2-2 is for a sort work file. On this statement, you typically code just the Assign clause to identify the file just as you would on a mainframe. If you're using Net Express, however, you can include the File Status clause, which can also be coded as Sort Status. Then, you can use the File Status field to check that the sort worked properly.

Typical Select statements for indexed files

A Select statement for an indexed file that's accessed sequentially

```
SELECT INVMAS  ASSIGN TO "c:\cobol\data\invmast.dat"
              ORGANIZATION IS INDEXED
              ACCESS IS SEQUENTIAL
              RECORD KEY IS IM-ITEM-NO.
```

A Select statement for an indexed file that's accessed randomly

```
SELECT INVMAS  ASSIGN TO "c:\cobol\data\invmast.dat"
              ORGANIZATION IS INDEXED
              ACCESS IS RANDOM
              RECORD KEY IS IM-ITEM-NO.
```

A Select statement for an indexed file with an alternate index

```
SELECT OPENITEM ASSIGN TO "c:\cobol\data\openitem.dat"
              ORGANIZATION IS INDEXED
              ACCESS IS SEQUENTIAL
              RECORD KEY IS OI-INVOICE-NUMBER
              ALTERNATE RECORD KEY IS OI-CUSTOMER-NUMBER
              WITH DUPLICATES
              FILE STATUS IS OPENITEM-FILE-STATUS.
```

Typical Select statements for relative files

A Select statement for a relative file that's accessed sequentially

```
SELECT INVMAS  ASSIGN TO "c:\cobol\data\invmastr.dat"
              ORGANIZATION IS RELATIVE
              ACCESS IS SEQUENTIAL.
```

A Select statement for a relative file that's accessed randomly

```
SELECT INVMAS  ASSIGN TO "c:\cobol\data\invmastr.dat"
              ORGANIZATION IS RELATIVE
              ACCESS IS RANDOM
              RELATIVE KEY IS INVMAS-RR-NUMBER.
```

A typical Select statement for a sort work file

```
SELECT SORTWORK ASSIGN TO "c:\cobol\data\sortwk01.dat".
```

Description

- The syntax of the Select statement for indexed and relative files in Net Express is the same as it is for a mainframe. The only difference in the way you code this statement is how you code the system name.
- A Select statement for an indexed file that contains alternate indexes must contain an Alternate Record Key clause for each alternate index. On a mainframe, this clause is required only if you want to access the file by the alternate index or you want to update the alternate index.
- When you code a Select statement for a sort work file in Net Express, you can include a File Status or Sort Status clause. This works just like it does for other files. The possible status codes for a sort work file are 00 for successful completion, 30 for a permanent I/O error, and 9x for various operating system errors.

Figure 2-2 How to code Select statements for indexed, relative, and sort work files

How to display and work with the data in a file

Net Express provides a tool called the *Data File Editor* that you can use to display and work with the data in a variety of files. In particular, you can use this tool to display and work with the data in sequential, relative, and indexed files. You'll see how to do that in the topics that follow. You'll also see how to use a text editor or a word processing program to display and work with a print file.

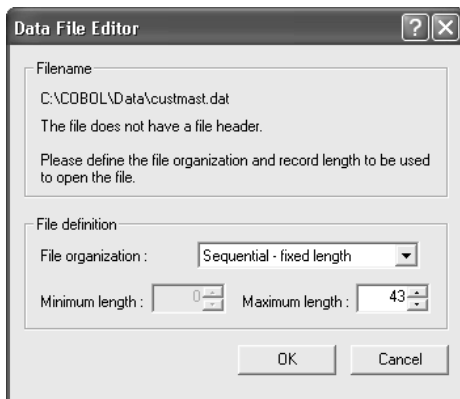
How to display and work with the data in a sequential file

Figure 2-3 shows how to display and work with the data in a sequential file. To display a file, use the File→Open command and then identify the file in the Open dialog box that's displayed. Then, if you're opening the file for the first time, the Data File Editor dialog box shown at the top of this figure is displayed. This dialog box lets you select the file organization and the length of the records in the file. For a sequential file, you can select either Sequential or Line Sequential for the organization.

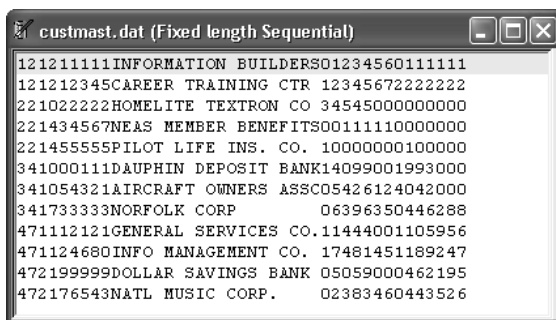
When you click on the OK button from the Data File Editor dialog box, a message is displayed indicating that you can avoid having to enter the file details in the future by saving them in a *profile*. If you click on the Yes button, the file details are saved in a file with the same name as the data file but with an extension of *pro*.

After you provide the appropriate details, the file is displayed in a Data File Editor window like the one shown here. You can use this window to insert, edit, and modify the records in the file as described in the figure. Then, when you close the file, you're asked to confirm the changes.

The Data File Editor dialog box



A sequential file displayed in a Data File Editor window



Description

- Net Express provides a *Data File Editor* that makes it easy to display and work with the data in a file.
- To display a file in the Data File Editor window, open it using the File→Open command. Unless you have saved information about the file organization and record length in a *profile*, a Data File Editor dialog box is displayed that prompts you for this information.
- When you complete the Data File Editor dialog box, you're asked if you want to save this information in a profile, which has the same name as the file and an extension of *pro*. If you create a profile, Net Express will automatically use the information it contains the next time you open the file.
- To edit the data in a record, type over the existing data. To delete a record, right-click on it and select the Delete Record command. To insert a record, right-click on an existing record and select the Insert Record Before or Insert Record After command. Then, enter the data for the new record.
- By default, a warning is displayed each time you attempt to edit or delete a record. If that's not what you want, you can select the Switch update warning off option in the dialog box that's displayed. When you close the file, you're prompted to save or cancel the changes.

Figure 2-3 How to display and work with the data in a sequential file

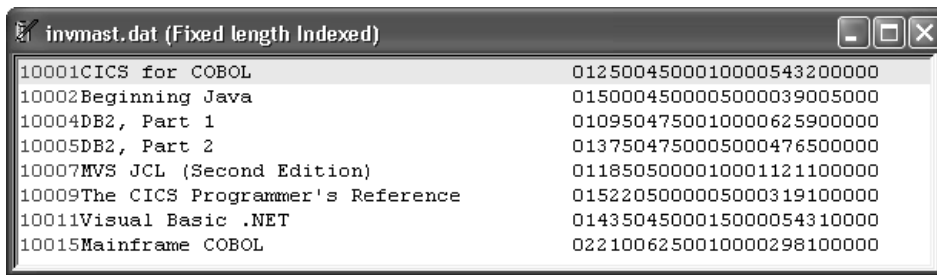
How to display and work with the data in an indexed or relative file

Figure 2-4 shows a Data File Editor window with an indexed file displayed and a Data File Editor window with a relative file displayed. Note that when you open an indexed file, the Data File Editor dialog box isn't displayed. That's because an indexed file includes a header record that contains the information needed to display the file. You have to provide that information for a relative file, though. Or, you can save it in a profile just as you can for a sequential file.

To edit or delete a record in an indexed or relative file, you use the same techniques you use for working with a sequential file. When you add a record to an indexed file, however, you're first asked to enter the key for the record. Then, you're returned to the Data File Editor window where you can enter the data for the rest of the record. Similarly, when you add a record to a relative file, you must first provide the relative record number for the record.

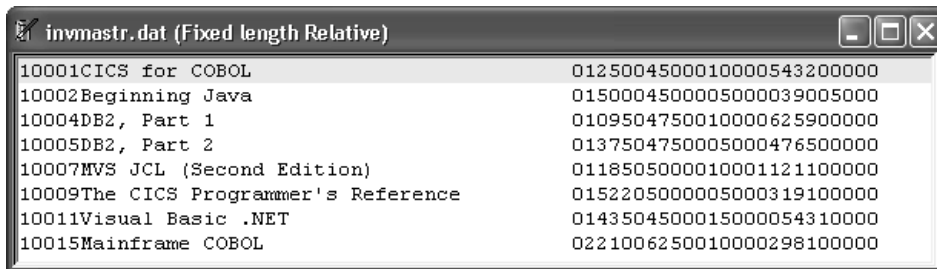
Unlike a sequential file, the changes that you make to an indexed or relative file are saved immediately. Net Express reminds you of that each time you open an indexed or relative file. However, you can turn this message off if you like.

An indexed file displayed in the Data File Editor window



Key	Description	Value
10001	CICS for COBOL	0125004500010000543200000
10002	Beginning Java	0150004500005000039005000
10004	DB2, Part 1	0109504750010000625900000
10005	DB2, Part 2	0137504750005000476500000
10007	MVS JCL (Second Edition)	0118505000010001121100000
10009	The CICS Programmer's Reference	0152205000005000319100000
10011	Visual Basic .NET	0143504500015000054310000
10015	Mainframe COBOL	0221006250010000298100000

A relative file displayed in the Data File Editor window



Key	Description	Value
10001	CICS for COBOL	0125004500010000543200000
10002	Beginning Java	0150004500005000039005000
10004	DB2, Part 1	0109504750010000625900000
10005	DB2, Part 2	0137504750005000476500000
10007	MVS JCL (Second Edition)	0118505000010001121100000
10009	The CICS Programmer's Reference	0152205000005000319100000
10011	Visual Basic .NET	0143504500015000054310000
10015	Mainframe COBOL	0221006250010000298100000

Description

- If you open a relative file and a profile isn't available for that file, the Data File Editor dialog box is displayed so you can specify the file organization and record length.
- An indexed file includes a header record that contains information about the file. Because of that, the Data File Editor dialog box isn't displayed when you open an indexed file.
- When you open an indexed or relative file, Net Express warns you that any changes you make will be applied immediately. To turn this message off, select the Do not show this message again option in the dialog box that's displayed.
- To edit the data in an indexed or relative file, type over the existing data. To delete a record, right-click on it and select the Delete Record command.
- To insert a record into an indexed file, right-click anywhere in the window and select the Insert Indexed Record command. You will be prompted for the key value (see figure 2-7). Then, you can enter the rest of the data for the record.
- To insert a record into a relative file, right-click anywhere in the window and select the Insert Relative Record command. You will be prompted for the relative record number (see figure 2-8). Then, you can enter the data for the record.
- By default, a warning is displayed each time you attempt to edit or delete a record. If that's not what you want, you can select the Switch update warning off option in the dialog box that's displayed.

Figure 2-4 How to display and work with the data in an indexed or relative file

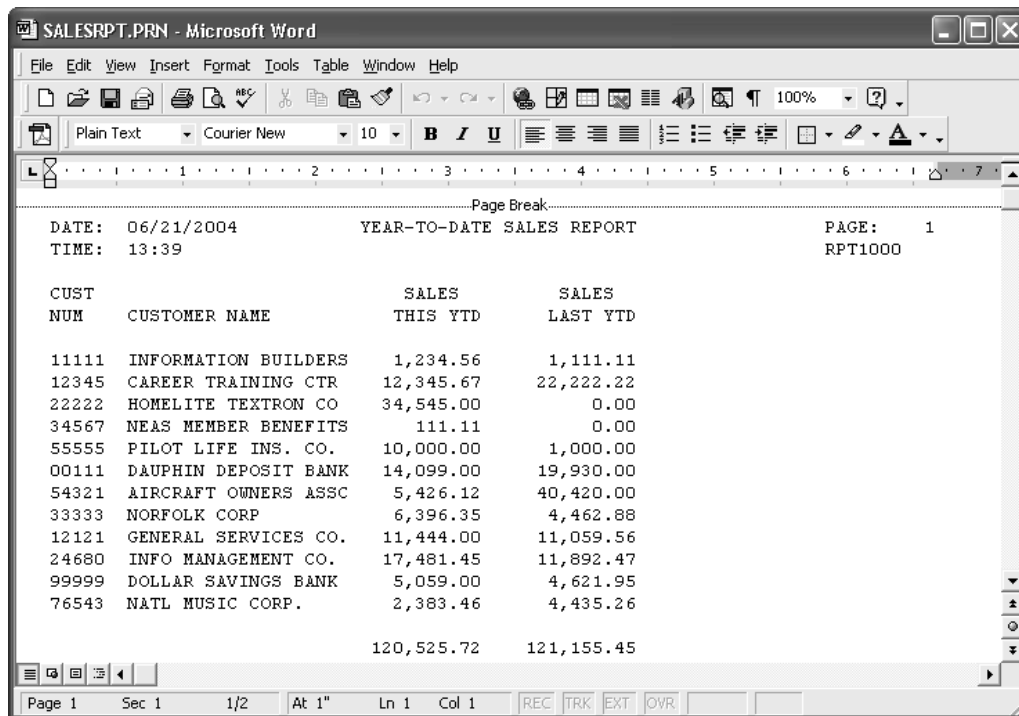
How to display and work with the data in a print file

Because a print file is a sequential file, you can display it in a Data File Editor window just like any other sequential file. If you want to change the layout or format of a print file, however, you're more likely to use a word processing program like Word or a text editor like NotePad. In figure 2-5, for example, you can see a sales report displayed in Word.

If a print file has long lines, you may need to format the report before it looks the way you want it to. For instance, you may want to change the layout from portrait to landscape or reduce the size of the font. Once you've got it the way you want it, you can review it on the screen or print it.

In this figure, you can see that the report starts with a page break because the program skips to the next page before it prints the first line. To save paper, then, you can delete the page break before you print the report. For many of your unsuccessful test runs, though, you won't need to print the report at all.

A print file displayed in Word



Description

- To review or print the data in a print file, you can open the file with your word processing program or with NotePad. To make the data more readable, you may want to change the page layout to landscape, change the margins, or change the font or font size. Then, you can review the output without printing it, or you can print it.
- You can also open a print file in Net Express. However, it doesn't provide the same flexibility for formatting a print file.

Figure 2-5 How to display and work with print files

How to create test files

In addition to using the Data File Editor to display and work with data files, you can also use it to create data files. That's particularly useful when you need to create a small file that you can use to test a program. In the topics that follow, you'll learn how to create sequential, relative, and indexed files.

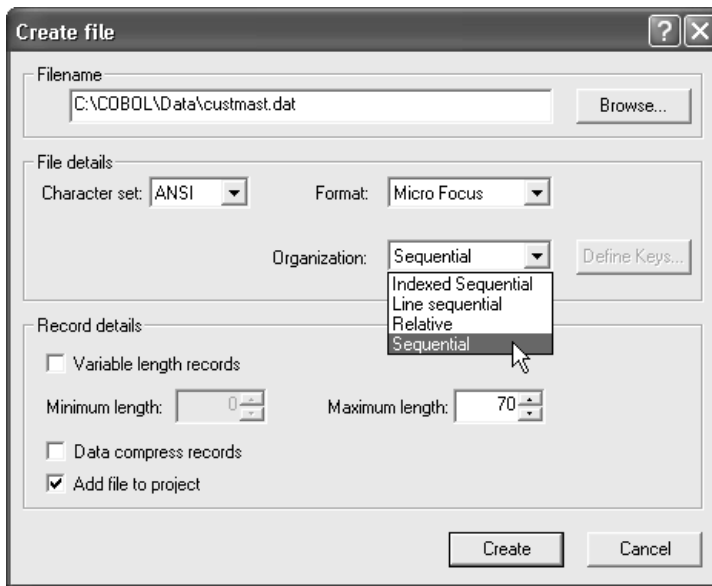
How to create a sequential file

Figure 2-6 shows the dialog box you use to create a file. As you can see, this dialog box lets you specify the path and file name of the file, the file's organization, and the length of the records in the file. To create a sequential file, you can select either the Sequential or Line sequential option.

Most of the other options should be set the way you want them. If you want to create a file with variable-length records, however, you'll need to select the Variable length records option. Then, you'll need to enter both the minimum length and maximum length of the records.

After you complete the Create file dialog box, another dialog box is displayed that asks if you want to create a profile for the file. After you respond to this dialog box, a Data File Editor window is displayed. Then, you can use the techniques you learned in figure 2-3 to insert, edit, and delete records.

The Create file dialog box



How to create a sequential file

1. To create a new file, select the File→New command to display the New dialog box. Then, select Data File and click the OK button to display the Create file dialog box.
2. Enter the path and name for the file you want to create, select Sequential or Line sequential organization, and enter the maximum length for the records in the file.
3. Click on the Create button to create the file and display an empty Data File Editor window. Click on the Yes button when prompted to save the file information in a profile.
4. Use the techniques presented in figure 2-3 to insert, edit, and delete records.

Notes

- By default, the file you create has fixed-length records. If you want to create a file with variable-length records, select the Variable length records option and then enter both the minimum and maximum record lengths.
- By default, the file you create is added to the current project if one is open. That makes it easy to open and work with the file. If that's not what you want, you can remove the check mark from the Add file to project option. Or, to remove the file from the project later, right-click on it in the right pane of the project window and select the Remove from project command.

Figure 2-6 How to create a sequential file

How to create an indexed file

To create an indexed file, you select Indexed Sequential organization from the Create file dialog box. When you do, the Define Keys button is enabled. When you click on this button, a Key Information dialog box like the one shown in figure 2-7 is displayed.

The first time the Key Information dialog box is displayed, it indicates that no key information is available. That's because no keys have been defined for the file. To define the primary key for the file, you click on the Insert Key button that appears in the Manipulate Keys and Components group near the bottom of the dialog box. Then, you enter the offset and length of the key in the Key Component group. In this figure, for example, you can see that the primary key will occupy the first five bytes of each record. Notice that the values you enter are also reflected in the tree at the left side of the dialog box. This tree shows all the keys that are defined for the file and the components of those keys.

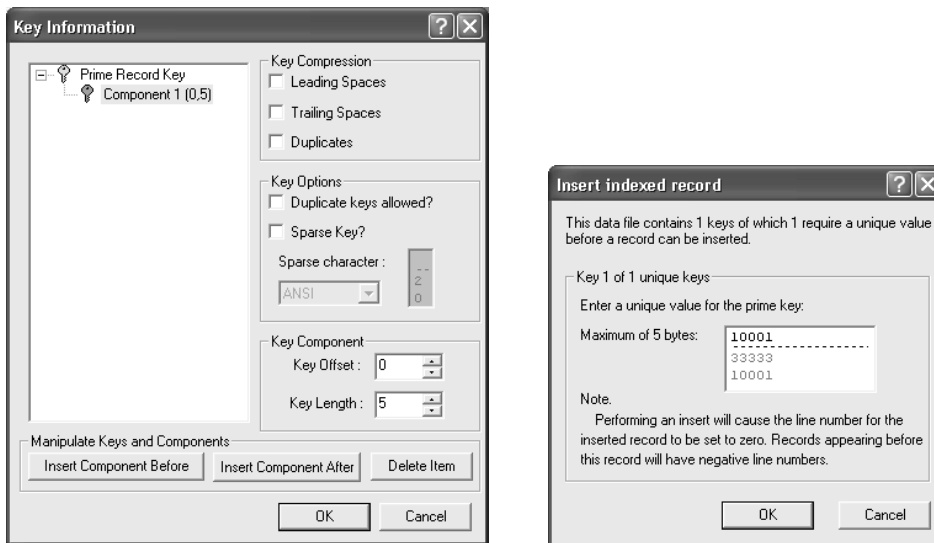
In most cases, that's all you need to do to define the primary key for a file. However, if the primary key is a composite key—that is, a key that consists of two or more fields—you'll need to add each field as a separate component. To do that, you select an existing component and then use the Insert Component Before or Insert Component After button to insert and define another component.

You can also use the Key Information dialog box to define an alternate key. To do that, select the Prime Record Key item in the tree at the left side of the dialog box. Then, click on the Insert Key Before or Insert Key After button that appears in the Manipulate Keys and Components group and enter the appropriate information.

For most files, you won't need to change any of the other options in the Key Information dialog box. If you define an alternate key, however, you may need to select the Duplicates option in the Key Compression group. That's the case if you want to allow more than one record in the file to have the same alternate key value.

After you complete the Key Information dialog box, you are returned to the Create file dialog box. Then, you can click on the Create button to create the file and display an empty Data File Editor window. From this window, you can use the Insert Indexed Record command that's displayed when you right-click in the window to add a record to the file. When you select this command, the Insert indexed record dialog box shown in this figure is displayed for each key that's defined for the file. This dialog box lets you enter the key values. After you do that, you're returned to the Data File Editor window, where the record appears in primary key sequence. Then, you can enter the remaining data for the record.

The Key Information and Insert indexed record dialog boxes



How to create an indexed file

1. Display the Create file dialog box. Then, enter the path and file name for the file you want to create, select Indexed Sequential organization, and enter the maximum length for the records in the file.
2. Click on the Define Keys button to display the Key Information dialog box, and then click on the Insert Key button. Enter the key offset and length for the first component of the key, and check any other options that apply.
3. If the key consists of additional components (a composite key), use the Insert Component Before and Insert Component After buttons to define the components.
4. Click on the OK button to return to the Create file dialog box. Then, click on the Create button in that dialog box to create the file and display it in a Data File Editor window.
5. For each record to be added to the file, right-click in the Data File Editor window and select the Insert Indexed Record command. Then, enter the key value for the record in the Insert indexed record dialog box that's displayed and click on the OK button to return to the Data File Editor window. Enter the rest of the data for the record.

Notes

- If you select the Prime Record Key item in the Key Information dialog box, the Insert Component Before and Insert Component After buttons change to Insert Key Before and Insert Key After. You can use these buttons to insert and define alternate keys for the file.
- If you define a file with alternate keys, you will be prompted to enter the key values after you enter the value of the primary key.

Figure 2-7 How to create an indexed file

How to create a relative file

To create a relative file, you select Relative organization from the Create file dialog box and enter the other required information. Then, when you click on the Create button, an empty Data File Editor window is displayed. To add a record to the file, you right-click in this window and select the Insert Relative Record command. When you do, the Insert relative record dialog box shown in figure 2-8 is displayed. This dialog box prompts you to enter a relative record number for the record. After you do that, you're returned to the Data File Editor window where you can enter the data for the record.

When you insert a record, the record appears in the correct sequence in the Data File Editor window according to its relative record number. For example, the relative record number for the records in the relative file shown in figure 2-4 are calculated by subtracting 10,000 from the item number that appears in the first five bytes of each record. So if you insert a record with a relative record number of 13, it will appear after the record that has a relative record number of 11, which is the record with item number 10011. Because the relative record number itself is not displayed, this can be confusing if you don't know how the relative record number is calculated or if the calculation is complex.

The Insert relative record dialog box



How to create a relative file

1. Display the Create file dialog box. Then, enter the path and file name for the file you want to create, select Relative organization, and enter the maximum length for the records in the file. Click on the Create button to create the file and display it in a Data File Editor window.
2. For each record to be added to the file, right-click in the Data File Editor window and select the Insert Relative Record command. Then, enter the relative record number for the record into the Insert relative record dialog box that's displayed, and click on the OK button to return to the Data File Editor window. Enter the data for the record.

Figure 2-8 How to create a relative file

How to use copy members

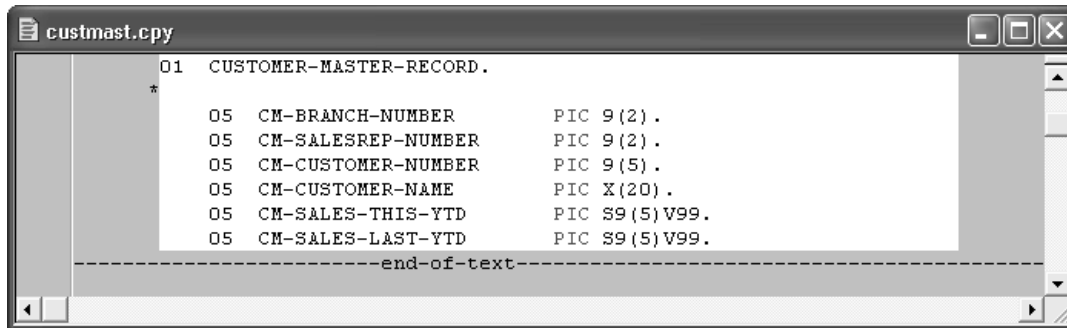
When you develop COBOL programs using Net Express, you can store code that's used by more than one program in files called *copy files* or *copy members*. Then, you can use the Copy statement to copy the code into each program that needs it. That saves you coding time, reduces errors, and simplifies maintenance.

How to create a copy member

Because a copy member contains COBOL code, you create it using the same techniques that you use to create a COBOL program. This technique is summarized in figure 2-9. First, you start a new project. Then, you add a new file for the copy member, and you give it an extension of *cpy* to identify it as a copy member. Finally, you enter the code for the copy member into the new file. The copy member shown at the top of this figure, for example, contains the record description for a customer master file.

When you use copy members, it's common to store two or more related copy members in the same project. In that case, the project can be thought of as a *copy library* or *source statement library*. Using a project this way can help keep your copy members organized.

A copy member that's stored in the file named c:\cobol\copy\custmast.cpy



How to create a copy member

1. Start a new project, or open an existing project that contains copy members related to the one you're creating.
2. Use the Add files dialog box to add a file to the project for the copy member. This file should have an extension of *cpy*.
3. Enter the code for the copy member just as you would for a COBOL program.

Description

- A *copy member*, or *copy file*, contains COBOL source statements that can be copied into a COBOL source program.
- With Net Express, a copy member is just a file that has *cpy* as its extension.
- To keep copy members together, you can store them in a project. Then, you can think of that project as a *copy library*, or *source statement library*.

Figure 2-9 How to create a copy member

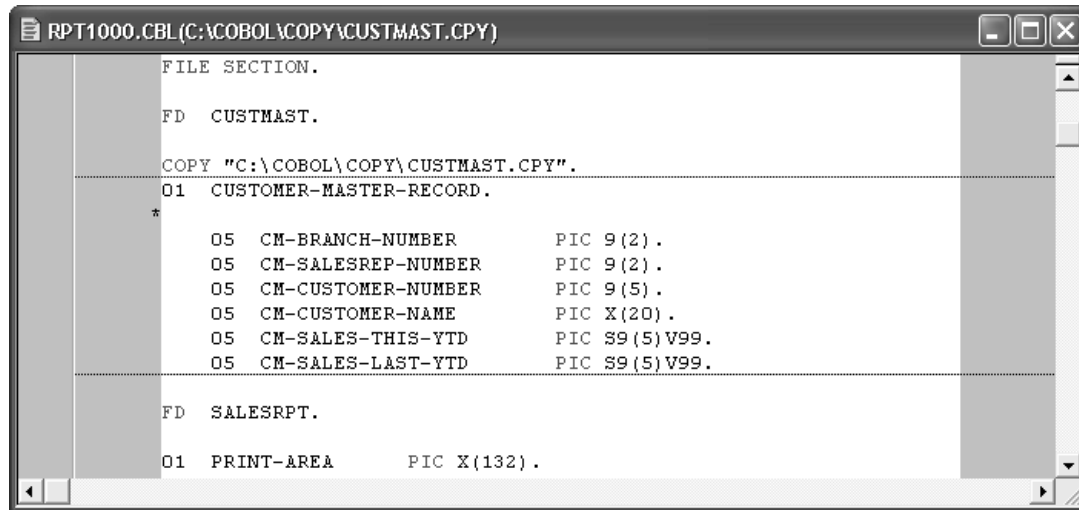
How to code the Copy statement

After you create a copy member, you can use the Copy statement to copy its contents into any of your programs. Figure 2-10 shows you how. The example shown here is for the copy member you saw in the previous figure.

If you want to display the source statements in a copy member, you can do that using the File→Copyfile→Show command as described in the figure. Then, the copy member appears as shown at the top of this figure. To hide the copy member again, you can use the File→Copyfile→Hide command. You can also use the Hide All Copyfiles button in the toolbar to hide and display all the copy members in a program.

You can also modify a copy member from within a program that uses it. To do that, just display the copy member and make the appropriate changes. Then, use the File→Copyfile→Save command to save the copy member.

How a copy member looks when displayed in a source program



The basic syntax of the Copy statement

```
COPY "file-name"
```

A typical Copy statement

```
COPY "c:\cobol\copy\custmast.cpy".
```

Description

- To use a copy member in a program, you code a Copy statement. Then, when you compile the program, the copy member is copied into the source program.
- To display a copy member used in a program, place the cursor in the line that contains the Copy statement and then select the File→Copyfile→Show command. To hide the copy member, place the cursor in the line that contains the Copy statement or anywhere in the copy member and select the File→Copyfile→Hide command.
- You can also display and hide all the copy members in a program using the Hide All Copyfiles button in the toolbar.
- While a copy member is displayed, you can modify it any way you like. Then, you can use the File→Copyfile→Save command to save the modifications.

Figure 2-10 How to use the Copy statement

Perspective

In this unit, you learned how to use Net Express to develop programs that work with files. Keep in mind, however, that many of the COBOL statements you use to work with files have not been presented here. That's because you code many of these statements the same way whether the file is stored on a mainframe or a PC. So if you want to learn more about working with files, you should read section 3 of *Murach's Mainframe COBOL*.

Summary

- Net Express provides for three types of sequential files. A *record sequential file* consists of a single string of characters. A *line sequential file* consists of records that end with a delimiter. And a *printer sequential file* consists of print records.
- The *system name* that you code for a file on a PC consists of a path and a file name.
- Net Express provides a tool called the *Data File Editor* that lets you display and work with the data in any type of data file. You can also use the Data File Editor to create new data files.
- You typically use a word processor or text editor to display and work with printer files. Then, you can format the output so it looks the way you want it to.
- You can use Net Express to create *copy members* that contain COBOL source statements. Then, you can use the Copy statement to copy those statements into any program that needs them.
- You can store related copy members in a single project. Then, the project can be thought of as a *copy library*, or a *source statement library*.

Terms

record sequential file
sequential file
line sequential file
printer sequential file
printer file
print file
Data File Editor
profile
copy member
copy file
copy library
source statement library

Objectives

- Given the specifications for a program that uses a sequential, indexed, relative, or sort work file, use Micro Focus Net Express to develop the program.
- Use the Data File Editor to display the data in any data file. Then, use the Data File Editor to insert, edit, and delete records.
- Given the specifications for a record sequential, line sequential, indexed, or relative file, use the Data File Editor to create the file.
- Given the specifications for a copy member, develop the copy member and then use it in a program.
- Describe the difference between a record sequential and a line sequential file.
- Describe the difference between a record sequential and a printer sequential file.