MACRO TUTOR
By Gordon McComb

About Macro Tutor

Macro Tutor is a short and concise introduction to WordPerfect 5.1 macros. The material in this document is drawn from the book WordPerfect 5.1 Macros and Templates, by Gordon McComb (published by Bantam Computer Books).

More About WordPerfect 5.1 Macros and Templates

WordPerfect 5.1 Macros and Templates, published by Bantam Computer Books, is a compendium of ideas, tricks, and shortcuts for using WordPerfect's macro feature. The book contains 400 ready-to-run macros and other files, including a complete letter writing system with 88 pre-written business letters. All macros and support files are provided on two diskettes bundled with the book. WordPerfect 5.1 Macros and Templates is written for the intermediate to advanced WordPerfect user, but does not require previous knowledge of the macro language.

WordPerfect 5.1 Macros and Templates is divided into 21 chapters and three appendices. Macro Tutor is an adaptation of portions of the first four chapters.

PART ONE -- MACRO BASICS
Chapter 1 -- Macro Primer
Chapter 2 -- Adding Power to Macros
Chapter 3 -- Editing Macros
Chapter 4 -- Learning the Macro Programming Language
Chapter 5 -- Making More of Macros
Chapter 6 -- Enhancing Keyboard Recorded Macros
Chapter 7 -- Creating Keyboard Layouts
Chapter 8 -- Merging With WordPerfect
Chapter 9 -- Creating Menu Systems

PART TWO -- MACRO APPLICATIONS
Chapter 10 -- Letters, Memos, and Correspondence
Chapter 11 -- Law Practice
Chapter 12 -- Desktop Publishing
Chapter 13 -- Invoices, Statements, and Receipts
Chapter 14 -- Mailing Labels and Envelopes
Chapter 15 -- Resumes
Chapter 16 -- Document Preparation
Chapter 17 -- Macro Potpourri

PART THREE -- GOING FURTHER WITH MACROS
Chapter 18 -- Macro Strategies
Chapter 19 -- For Pros Only: Advanced Techniques
Chapter 20 -- Macro Debugging

PART FOUR -- THE LETTER MAKER MACRO SYSTEM
Chapter 22 -- Using Letter Maker

Appendix A -- How to Use the Applications Disk
Appendix B -- Disk Contents
Appendix C -- Macro Codes

WordPerfect 5.1 Macros and Templates, by Gordon McComb, is available at most book and software stores, including B. Dalton, Waldenbooks, Software Etc., and Egghead Software. The book can also be ordered directly from the author, for $39.95 postpaid (California residents please add 6.75% sales tax). Send check or money order to:

Gordon McComb
2642 Hope St.
Oceanside, CA 92056

CHAPTER 1
MACRO PRIMER

Imagine turning on a tape recorder in your computer to memorize everything you do on the keyboard. Press a series of keys to save a document on your hard disk, for example, and the computer remembers each key you pressed. Play back the recording, and the computer mimics your actions. That's the idea behind WordPerfect macros -- memorizing your keystrokes and playing them back at the touch of a button. Of course, macros go way beyond this simple definition. You will learn all about macros, and the various functions they can perform, throughout the course of this book.

For the sake of simplicity, WordPerfect macros can be divided into two categories:

. Keyboard recorded, where the keys you press are recorded in a special macro file. You play back the file and WordPerfect repeats the original recording.

. Programmed, where you use a programming language to instruct WordPerfect on what to do.

In this chapter, you'll learn what keyboard recorded macros are and how they are used. This chapter serves as an introduction only; more advanced macros techniques are covered in the following chapters.

WHEN TO USE MACROS

You can use macros for a wide variety of tasks. The way you use macros will depend on your application, and how often you do the same types of jobs. In any case, macros are used to simplify WordPerfect and cut down on the amount of typing you do. You can define a macro to perform any task you would normally perform with WordPerfect, but you should reserve macros for only those chores that you do on a fairly regular basis.

As a rule of thumb, you don't need the macro if it isn't used more once or twice a week (assuming you sit behind the computer every day). Unless the series of keystrokes is very long and involved, you're better off manually tapping it out on the keyboard. Some exceptions to this "rule" apply, however, particularly if WordPerfect is used by persons with little or no computer training.

Using macros falls into four distinct categories:

. To automate command key sequences.
. To memorize passages of commonly used text.
. To simplify a series of formatting instructions.
. To program WordPerfect for use by others.

Let's take a closer look at each category and examine the special relationship each has with WordPerfect macros.

To Automate Command Key Sequences

WordPerfect employs a complex structure of menus, commands and options using the computer's function keys. Some of the commands, such as underlining, document saving, and spelling, provide only one or two options and their simplicity does not require macros.

Other commands, however, have several layers of menus and using them requires you to choose the functions you want from a list that appears on the screen. If you find yourself choosing the same commands and options time and time again, you can automate the process by recording the keys as a macro.

To Memorize Passages Of Commonly Used Text

One common use of WordPerfect macros is memorizing passages of text for later playback. Instead of tapping out "WordPerfect" each time you write it, you record it as a macro and recall it at touch of a key. You can do the same thing with other text entries, including your name and address, telephone number, company name, and common phrases such as "Sincerely," "I look forward to your prompt reply" and "You just won $10,000 in our 'Welcome to the Neighborhood' sweepstakes!"

The length of the passage of text is unlimited -- although very long text blocks are better saved as a merge document. You can use macros to store "boilerplate" text for use in contracts, legal briefs, reply letters, and other documents. Instead of typing commonly used words and phrases, you record the text as a macro and insert them in your documents at any point you desire. Boilerplating allows you to customize your documents while saving time and effort.

An often overlooked advantaged of using macros to store frequently used text is that, because the characters are stored on disk in the exact form that you typed them, your accuracy and spelling improves. It's easy to overlook typographical errors in the return address, salutation, or closing of a letter. And imagine how embarrassing it is to end your letter "Sincereky, Rogerr Smth," instead of the proper "Sincerely, Roger Smith." Of course, the spelling checker built inside WordPerfect may catch some or all of these errors, but you may forget to check the letter before mailing it out. Perhaps a more serious error is entering the wrong return address, social security number, or phone number, items that the spelling checker does not review.

To Simplify a Series Of Formatting Instructions

WordPerfect's default format settings are useful for routine letter writing, but for more advanced documents you need to alter the format characteristics. Every element of appearance of WordPerfect documents is controlled by one or more of the computer's function keys. To change the margins, for example, you press:

. **[Shift]**-**[F8]**
. Option **2**
. Option **7**

You then enter the desired right and left margins. Depending on the margin settings you want, this operation requires a minimum of 10 keystrokes (including pressing the **[Enter]** key to accept the changes and return from the menus).

Granted, the keystrokes necessary to change the margin settings are not overly complicated, so if you modify the margins only occasionally, you can continue to plunk away manually at the keyboard. But if you find yourself frequently switching between common margin settings, you'll save a great deal of time by encoding the changes as a set of macros. With standard margin settings stored as a macro, you can reduce the number of keystrokes to two -- the **[Alt]** key and a single letter.

Some document types require you to constantly change the margins. If you are writing movie scripts, for example, you need different margin and tab settings for dialog and action sections. In the typical one hour TV script, you may change margins two or three hundred times. That equates to two or three thousand keystrokes that never appear in your script, and aren't likely to help win you that Emmy award. By recording the various margins as macros, you can quickly recall the proper settings by pressing just a few keys.

To Program WordPerfect For Use By Others

Not everyone is a whiz at WordPerfect. If others in your office use WordPerfect, and they have little or no experience with it, you can program the software with macros to make it easier to use. A good example is creating and printing a mailing list. Normally, this requires fairly in-depth knowledge of WordPerfect. But by creative use of macros, you can program the software to prompt the user for the desired information and nearly automate the entire process.

Even if you or others in your office are familiar with Wordperfect, macros can be used to simplify complicated tasks, particularly those that you may not often do. The macro stores all of the commands needed to complete a procedure, so you don't have to remember which buttons to press.

BASIC STEPS

There are five basic steps to defining a macro.

. Step 1: Press **[Ctrl]-[F10]** to begin the macro definition.

. Step 2: Name the macro in any of the following ways explained in the section below.

. Step 3: Describe the macro. Enter a line of text to describe the macro (up to 39 characters). The comment is visible only when editing the macro. The prompt **Define Macro** now appears on the screen.

. Step 4: Define the actual macro by entering the keystrokes you want recorded. While defining the macro, the prompt **Macro Def** flashes on the screen.

. Step 5: Press **[Ctrl]-[F10]** once more to complete the macro definition.

While defining a macro, WordPerfect will execute your commands. This helps you better follow the course of the macro, but you may also inadvertently change your document in way you don't want. You can create macros without disturbing current work by switching to Doc 2 (press the **Switch** key). After you complete the macro, switch back to the main document or close Doc 2 (press **Exit**, **N**, **N**.)

TYPES OF MACROS

WordPerfect creates four types of macros:

. Alt-key macros
. Named macros
. Remapped-key macros
. Temporary macros

Alt-key Macro

WordPerfect can store up to 26 Alt-key macro definitions -- one for each letter key. They are termed Alt-key macros because of the way they are recalled: you press the **[Alt]** key and a letter key.

To define an Alt-key macro:

	Key Sequence	What it Does
1.	**[Ctrl]**-**[F10]**	Starts macro recording.
2.	**[Alt]**-{key}	Names macro with Alt-key combination (such as **[Alt]-A** or **[Alt]-F**).
3.	{description} **[Enter]**	Describes the macro.
4.	{macro steps}	Defines the steps of the macro (enter keystrokes to be recorded here).
5.	**[Ctrl]**-**[F10]**	Stops macro recording; saves macro file on current default disk.

To replay an Alt-key macro, press the **[Alt]** key and the desired letter key simultaneously. For example, if you recorded a macro that changes the right margin from 74 to 55 as **[Alt]**-**M**, depress the **[Alt]** and **M** keys together.

Alt-key macros are stored on disk and can be accessed any time, even during subsequent sessions with WordPerfect, regardless of the document you are editing (with some word processors, macros are tied to the current document). Alt-key macro files can be copied, deleted, and manipulated like any other WordPerfect file.

WordPerfect automatically names the Alt-key macro file for you. You can easily identify Alt-key macro files by their name and .WPM extension (macros made with version 4.2 of WordPerfect, which cannot be used in version 5.1 without conversion, have the extension .MAC). The name is always "ALT" plus a single letter key.

Examples:
ALTA.WPM -- for the **[Alt]** and **A** key combination.
ALTD.WPM -- for the **[Alt]** and **D** key combination.

Named Macro

The Alt-key macros are the easiest to use because you can replay them by pressing just two keys (the **[Alt]** key and a letter key). You should record the most common procedures as Alt-key macros. But any serious WordPerfect user will quickly run out of Alt-key combinations. As an alternative, you can save macros under a name you provide. Once saved, you recall the macro by pressing **[Alt]**-**[F10]**, the Macro key, then entering the name of the macro you want.

To define a named macro:

	Key Sequence	What it Does
1.	**[Ctrl]**-**[F10]**	Starts macro recording.
2.	{name} **[Enter]**	Names macro using any name with two to eight characters.
3.	{description}	Describes macro.

	[Enter]	
4.	{<u>macro</u> <u>steps</u>}	Defines the steps of the macro (enter keystrokes to be recorded here).
5.	**[Ctrl]**-**[F10]**	Stops macro recording; saves macro file on disk.

To replay a named macro:

	Key Sequence	**What it Does**
1.	**[Alt]**-**[F10]**	Execute play back macro command.
2.	{<u>name</u>}	Defines name of macro to use.
3.	**[Enter]**	Initiates macro playback.

As with Alt-letter macros, named macro files can be identified by their .WPM extension.

Examples:
SALUTE.WPM -- for a salutation macro for use at the start of letters.
LET-HEAD.WPM -- for a letterhead macro.
MERGE.WPM -- for a merge print macro.

Macro naming conventions are the same as with DOS files. Macro names may contain from one to eight characters. You may use any combination of letters or numbers in the file name as well as most symbols except the following:
. Any control character, including Escape and Delete.
. A space.
. The characters ^ + = / \ [ ] " ; : ? * < > |
. A period (WordPerfect automatically adds the period and WPM file extension; you don't need to do it yourself.

Remapped-key Macros

WordPerfect allows you to reassign (or "remap") nearly every key on the keyboard. For example, you can change the standard "QWERTY" keyboard layout to the Dvorak layout by re-identifying the alphabet keys. The re-assigned keys may include multiple keystrokes. For instance, the **[Shift]**-**[F7]** key, which normally calls up the print menu, might be re-mapped to call up the Print menu, select the Draft Quality option, then print the entire document. In this way, the remapped keys behave like macros. In fact, you edit and program the keys the same way as you do macros.

Keyboard remapping uses the keyboard layout option in the Setup menu. The exact procedure for key remapping, which is an advanced topic, is detailed in Chapter 7.

Temporary Macro

Not all macros need to be stored for future use. You may have occasion to create a temporary macro that you use for the current writing and editing session only. The advantage of a temporary macro is that you won't clutter up your disks with a macro you may create and use only on one special occasion.

WordPerfect 4.2 allowed you to make up to 27 temporary macros, but version 5.1 permits only one. Temporary macros are assembled in nearly the same manner as permanent Alt-key macros. To define a temporary macro:

	Key Sequence	What it Does
1.	**[Ctrl]**-**[F10]**	Starts macro recording.
2.	**[Enter]**	Creates temporary macro.
3.	{<u>macro</u> <u>steps</u>}	Defines the steps of the macro (enter keystrokes to be recorded here).
4.	**[Ctrl]**-**[F10]**	Stops macro recording; saves macro file on disk.

To play back the temporary macro:

	Key Sequence	What it Does
1.	**[Alt]**-**[F10]**	Invoke play back macro command.
2.	**[Enter]**	Defines name of macro to use.

While the **[Enter]**-key macro is technically "temporary," WordPerfect records it on the disk as WP{WP}.WPM.

FINDING MACROS ON THE DISK

Macros can be on any disk and any directory. However, if the macro file is not on the current (or "default") drive and directory, WordPerfect will not be able to find it. The easiest to access a macro stored on a disk or directory other than the current one is to explicitly provide the drive and/or the directory.

Examples:

. **C:macros\dblspace** -- Finds the macro DBLSPACE.WPM (for double space) in the macros directory on drive C:.

. **A:dblspace** -- Finds the macro DBLSAPCE.WPM on drive A:.

. **macros\format\dblspace** -- Finds the macro DBLSPACE.WPM on the current drive in the macros and format subdirectories.

You may omit the drive and directory if you have told WordPerfect where to look for the macro files. The Location of Files option in the Setup menu allows you to identify a regular repository for your macros files. As you become a proficient user of macros, you will want to take advantage of this feature. It will make storing and finding macros much easier if you keep them on a separate disk or in a subdirectory.

You can still keep some macros in the main WordPerfect disk and/or directory (the one that contains the WP.EXE file). If WordPerfect can't find the macro it needs in the spot specified in the Location of Files option, it checks back in the WP.EXE directory.

If you are not sure how to create subdirectories, consult your DOS manual or look it up in any book on MS-DOS. In addition, WordPerfect will create subdirectories for you if you define one after selecting **List** key and pressing the = sign. Enter the name of the new subdirectory, then answer **Y** to the prompt.

ERASING, REDEFINING, AND EDITING MACROS

You can create, editor, or replace macros using the **Macro Define** (**[Ctrl]**-**[F10]**) key. If you choose a name for a macro that already exists, you are given four choices.

- Cancel macro definition. To cancel the new definition, press the **Cancel** key **([F1]**).
- Erase the previous macro and start over. To Replace the previous macro, press **R** or **1**.
- Edit the existing macro. To Edit the macro, press **E** or **2**.
- Edit the macro description. To edit the description, press **D** or **3**.

If you choose to replace the existing macro, WordPerfect double checks your choice with a Yes/No prompt. Answer carefully. Choosing Yes will forever erase your previous macro. If you don't need an Alt-key or named macro any more, you can erase it from your disks using either DOS or WordPerfect, in the normal manner.

Macro editing is an advanced topic and is discussed in future chapters.

CANCELING A MACRO

You may cancel a macro in execution at any time by pressing **Cancel**. If that doesn't work, try pressing the **[Ctrl]-[Break]** keys. WordPerfect macros have the ability to "cancel" the Cancel key, rendering it inoperative, but the **[Ctrl]-[Cancel]** key will normally halt any macro, no matter how it was programmed.

Note that most macros are replayed very fast, and that a series of even 100 keystrokes may take only a moment to replay. You won't have to press the **Cancel** key before the macro finishes. Rather, canceling a macro is handy when the macro has been slowed down or when you want to exit from a loop in a programmed macro.

You may cancel a macro that you are defining by pressing the **Macro Define** key. That ends the macro but it also saves what you've done so far on the disk. If you don't want the macro, delete it from the disk or create a new macro using the same name (see the section above on Erasing And Redefining Macros).

CHAPTER 2
LEARNING ADVANCED MACROS

There are a number of advanced macro features that are helpful in everyday applications of WordPerfect. These advanced features are part of WordPerfect, and are documented in the WordPerfect manual, but they are worth reiterating here. Many more advanced macro features are covered later in this book and a good number of these are not documented in the WordPerfect manual. These special features are tips and tricks that stretch the macro feature to provide new and unusual functions.

PAUSING MACROS FOR USER INPUT

Macros can be paused anywhere during playback and instructed to await direct keyboard entry. Pausing is helpful when you want to automate a process but still need to provide variable information. You can used paused macros, for example, to generated invoices. The macro creates the general form of the invoice, while pausing to allow you to enter data.

Another example is automating a standard form letter. One method of creating form letters is to use the merge print feature of WordPerfect. The merging process is time consuming if you mail out only a few letters. Another approach is to create a macro where standard (boilerplate) text is inserted for you. The computer stops at certain spots in the letter to allow you to enter specific information, such as the current date, the name of the addressee, the salutation, and so forth. Each pause ends when you press Enter.

Follow these steps to add a pause during a macro:

	Key Sequence	What it Does
1.	**[Ctrl]**-**[F10]**	Begins a new macro definition.
2.	{<u>name</u>} **[Enter]**	Names the macro.
3.	<u>{description}</u> **[Enter]**	Describes the macro.
4.	{<u>macro</u> <u>steps</u>}	Defines the steps for the first portion of the macro.
5.	**[Ctrl]-[Page Up]** **1**	Indicates that you want to add a pause.
6.	**[Return]**	Enters a pause.
7.	{<u>macro</u> <u>steps</u>}	Defines the steps for the remainder of the macro.
7.	**[Ctrl]**-**[F10]**	Completes macro definition.

Though the steps outlined above show one pause between two "macro steps" segments, you can insert as many pauses as you like during the macro. Simply repeat steps 5 and 6 for each new pause. When you press the **[Ctrl]**-**[Page Up]** key during macro definition.

To use the paused macro, replay it as usual. When the pause is encountered, the macro halts and awaits text from the keyboard. Type the text and when finished, press **[Enter]**. The macro will continue and either end, or pause at the next stop. Always remember to press the **[Enter]** key when you are finished typing the text from the keyboard. This signals to WordPerfect that you are done with the entry and the macro continues.

> **IMPORTANT NOTE**: While WordPerfect provides a "Macro Def" signal (in the lower-left corner of the screen) when you are defining a macro, it offers no such indicator when replaying a macro. You have no way of knowing if a macro is currently running. Adding pauses to a macro has the affect of momentarily freezing the macro and preventing it from continuing with the rest of the steps. <u>But the macro is still active.</u> If you're not sure whether a macro is still running, press the Macro Define (**[Ctrl]**-**[F10]**) key. If nothing happens, a macro is still in progress (when paused, you cannot define another macro). Future chapters will discuss other instances when a macro may appear to have stopped when in fact it's still running, and ways to provide your own "macro-in-progress" indicator.

Hands-On Pausing

Let's create a simple memo header using the pause feature. The memo will stop several times to allow you to enter the current date, the subject, and other pertinent information.

	Key Sequence	**What it Does**
1.	**[Ctrl]**-**[F10]**	Starts macro definition.
2.	**memo** **[Enter]**	Names the macro "memo."
3.	**Memo header** **[Enter]**	Describes the memo.
4.	**Date:**	Types Date:.
5.	**[Ctrl]-[Page Up]** **1 [Enter]**	Pauses for Date: input.
6.	**[Enter] [Enter]**	One blank line.
7.	**Subject:**	Types Subject:.
8.	**[Ctrl]**-**[Page Up]** **1 [Enter]**	Pauses for Subject: input.
9.	**[Enter] [Enter]**	One blank line.
10.	**To:**	Types To:.
11.	**[Ctrl]**-**[Page Up]** **1 [Enter]**	Pauses for To: input.
12.	**[Enter] [Enter]**	One blank line.
13.	**From:**	Types From:.
11.	**[Ctrl]**-**[Page Up]** **1 [Enter]**	Pauses for From: input.
12.	**[Enter] [Enter]** **[Enter]**	Two blank lines.
13.	**[Ctrl]**-**[F10]**	Ends macro definition.

When you replay the macro, it will write the Date:, Subject:, To, and From: fields, allowing you time to manually enter in the specific information. Press the **[Enter]** key each time you finish with a line.

HIDING DISPLAY DURING MACRO EXECUTION

Normally, WordPerfect starts each macro with the display turned off. With the display off, commands and menus aren't shown during macro execution, and the macro works much faster. However, the screen may remain blank if the macro pauses at a menu. If your macro will pause at a menu, you must turn the display on before actually reaching the menu. A good place to turn the display back on is immediately before selecting a WordPerfect function, such as Help of Format.

	Key Sequence	**What it Does**	
1.	**[Ctrl]**-**[F10]**	Begins a new macro definition.	
2.	{<u>name</u>}	Names the macro.	**[Enter]**
3.	{<u>description</u>}	Describes the macro.	
	[Enter]		
4.	{<u>macro steps</u>}	Defines the steps for the macro.	
5.	**[Ctrl]-[Page Up]**	Indicates that you want to control the display.	**2**
6.	**Y**	Turns the display on.	

....
Rest of macro

The display option can be turned on and off at any point within the macro. Press **[Ctrl]-[Page Up]**, then **2** to access the Display menu. Press **N** to turn the display off and **Y** to turn the display on.

> **IMPORTANT NOTE**: The **[Ctrl]**-**[Page Up]** key is multi-purpose, and behaves differently depending on the activity of WordPerfect. You can select the Display and Pause options (plus two others that will be covered in later chapters) only when defining a macro from the keyboard. Chapter 3 outlines the many uses of the **[Ctrl]**-**[Page up]** key.

AUTOMATICALLY EXECUTING OTHER MACROS

Some operations, like changing between single and double space, require only one macro. More advanced operations, such as preparing a mailing list from a data document, filling out an invoice, or creating a table of contents, may require two or more macros, or else one large programmed macro. When using separate macros, you invoke each one in turn to complete the task. You can execute these macros manually -- starting each one with the **[Alt]**-**[F10]** key, or include commands within one macro to automatically run another.

How can using separate macros be beneficial over stuffing everything in one large macro? One reason is for simplicity. Separate macros are easier to create and edit, especially for the beginning "macroist."

And, separate macros can be run either as a complete set, or independently. For example, you could create one macro that changes the margins, resets the tabs, and generates a header and footer. Or you could create several macros that each have a separate function. In the example,

. MACRO1 changes the margins.
. MACRO2 resets the tabs.
. MACRO73 generates a header and footer.

Used this way, individual macros allow you a great deal of flexibility. You use only those macros that you need for a particular task. If you need to generate a header and footer, use just the header/footer macro. If you need to change the margins and tabs, use the margin and tabs macros. Of course, you need to replay each macro separately. That means more keystrokes for you as you recall all the macros in the set. And, separate macros tend to clutter up your disk. With so many macros to keep track of, you may forget which macro does what function.

Chaining

Macros can start other macros. This is called <u>chaining</u>. With chaining, you can automatically step from one macro to the next. You start the first macro (the <u>root</u>) yourself. When the root macro finishes, it automatically calls up a second, or <u>sub</u>, macro. When that macro finishes, yet another one can be automatically started. You can continue these macro "links" to make a chain as long as you like.

Chaining relieves you of manually starting each macro in the set. But use chaining with caution. Macros refer to other macros in the chain by name, and if you delete a macro or alter its name, the chain will be broken. Keep a record of your macro chains to help you from inadvertently disturbing the links.

Follow these steps to make a macro chain. This example assumes you are creating a chained macro with two links; one called MACRO-A and the other called MACRO-B. For the sake of simplicity, we'll assume that MACRO-B already exists.

	Key Sequence	**What it Does**
1.	**[Ctrl]**-**[F10]**	Begins a new macro definition.
2.	**MACRO-A** **[Enter]**	Names the new macro MACRO-A.
3.	**Macro-A** **[Enter]**	Describes the macro as Macro-A.
4.	{<u>macro</u> <u>steps</u>}	Defines the steps for MACRO-A.
5.	**[Alt]**-**[F10]**	Starts sequence for MACRO-B chain.
6.	**MACRO-B** **[Enter]**	Indicates MACRO-B for chain.
7.	**[Ctrl]**-**[F10]**	Completes macro definition.

. Steps 1 through 4 define the macro for MACRO-A.
. Steps 5 and 6 start MACRO-B.
. Step 7 completes the macro definition and returns you to editing mode.

Note that MACRO-B is not actually replayed while you are creating MACRO-A. That prevents the keystrokes from MACRO-B from being included as part of MACRO-A. When you start MACRO-A, MACRO-B is automatically run. You can also start

MACRO-B normally, without using MACRO-A.

Creating A Macro Linker

Another way to chain separate macros into a set is to use a separate macro "linker." The linker is a "bare-bones" macro that does nothing but call other macros. It does this by the use of <u>nesting</u> a procedure whereby WordPerfect starts one macro but before it has ended, branches off to another macro. When the second macro finishes, WordPerfect returns to the first macro. You can write the linker using WordPerfect's macro language, but if the macro is not too complex, you can create it directly from the keyboard.

First, build the branch macros that you want to use; that is, create the macros that you will be nesting to. Let's say that you are creating a macro set that includes three branches:

1. Set margins to 1.5" on both sides.
2. Remove all tabs but one at the 3" mark.
3. Turn justification off and line numbering on.

Give each macro an Alt-letter name; nesting won't work otherwise (you can build nested macros without using Alt-letter macros, but it requires the use of the macro programming language). For our example, let's name these macros Alt-A, Alt- B, and Alt-C. Now create the linker macro, the one that will branch off to each of the three others in turn. Define the macro by pressing the **[Alt]**-**A**, **[Alt]**-**B**, and **[Alt]**-**C** keys in turn.

Hands-On Macro Nesting

So much for theory; now for practice. You can experiment with a simple macro linker by creating these four macros.

For Alt-A:

1.	**[Ctrl]**-**[F10]**	Starts macro definition.	
2.	**[Alt]**-**A**	Names the macro Alt-A	
3.	**Macro A** **[Enter]**	Describes the macro.	
4.	**This is Macro A**	Text for macro.	**[Enter]**
7.	**[Ctrl]**-**[F10]**	Ends macro definition.	

Repeat the same procedure for making the Alt-B and Alt-C macros, but change the names, description, and definition as required. When the the Alt-letter macros are finished, create the linker.

	Key Sequence	**What it Does**	
1.	**[Ctrl]**-**[F10]**	Starts macro definition.	
2.	**Linker** **[Enter]**	Names the macro ("Linker" in this example.	
3.	**Linker macro A,B,C**	Describes the macro.	**[Enter]**
4.	**[Alt]**-**A**	Nests in macro Alt-A.	
5.	**[Alt]**-**B**	Nests in macro Alt-B.	

6.	**[Alt]**-**C**	Nests in macro Alt-C.
7.	**[Ctrl]**-**[F10]**	Ends macro definition.

When you run the linker macro, it branches off to each Alt- letter macro in turn. The screen should read:

This is Macro A
This is Macro B
This is Macro C

Repeating Macros

You may repeat a macro as many times as you like using the **[Esc]** key. Before executing the macro, use the **[Esc]** key to indicate the number of times you want the macro repeated. For example, to repeat a macro named BOX five times, type the following:

	Key Sequence	What it Does
1.	**[Esc]**	Initiates repeat.
2.	**5**	Sets repeat to five times.
3.	**[Alt]**-**[F10]**	Invoke play back macro command.
4.	**BOX**	Indicates macro to run.
5.	**[Enter]**	Accepts the entry; macro is repeated five times.

Repeating Chains

Repeating chain macros use WordPerfect's search feature to automatically repeat a certain number of times. Repeating chain macros have an advantage over **[Esc]** key macros in that they will continue to work until the search results in a *not found* message. You don't have to manually calculate the number of times to repeat the macro.

Repeating chains use one macro that chains back to itself. Follow this procedure to create a repeating chain.

1. Begin defining the macro.
2. Enter the keystrokes and include a search of some type.
3. Call the macro from within itself.
4. End the macro.

Though repeating chains can be useful, most of their applications can be duplicated using WordPerfect's Replace command. The exception is when you must make a change that can't be entered into the replace string. For example, say that you need to reformat a document that's been imported from another word processor. The file has five empty spaces at the beginning of each paragraph instead of tabs. What's worse is that the document lacks soft returns. It's a fairly long document and you don't want to hunt out each hard return in the middle of a paragraph and remove it. After exchanging the empty spaces for tabs, create this macro.

	Key Sequence	What it Does
1.	**[Ctrl]**-**[F10]**	Starts macro definition.
2.	**softconv**	Names macro softconv.

[Enter]

3. **Convert to soft returns [Enter]**	Defines macro.
3. **[F2]**	Calls up search.
4. **[Tab]**	Inserts [Tab] code in search string.
5. **[F2]**	Starts search.
6. **[Alt]-[F4]**	Turns block on.
7. **[F2] [Tab] [F2]**	Searches for next [Tab].
8. **[Left] [Left] [Left]**	Moves cursor back three spaces (past [Tab] and [HRt] codes.
9. **[Alt]-[F2]**	Calls us replace; no confirm. **n**
10. **[Enter]**	Inserts [HRt] code in search string.
11. **[Alt]-[F2]· [Alt]-[F2]**	Starts replace for selected block
12. **[Alt]-[F10] softconv [Enter]**	Chains back to "softconv" macro.
13. **[Ctrl]-[F10]**	Ends macro definition.

. Steps 1 through 3 creates and defines the macro.
. Steps 3 through 5 searches for the [Tab] code at the beginning of each paragraph.
. Step 6 turns the block function on.
. Steps 7 and 8 search for the next [Tab] code (at the beginning of the following paragraph). The macro then steps back three spaces to end the block selection immediately before the last hard return code in the preceding paragraph (easier done that said!).
. Steps 9, 10, and 11 replaces the hard return codes with a space (WordPerfect automatically inserts soft returns).
. Step 12 repeats the softconv macro -- effectively looping the macro back on itself.

To use the macro, position the cursor at the beginning of the document and start the **softconv** macro. The macro will repeat itself until it finds no more [Tab] codes, at which time the "*not found*" message is displayed and the macro ends.

The *not found* message indicates to WordPerfect that a type of error has occurred. In turn, WordPerfect halts macro execution, in case there is something seriously wrong. Repeating chain macros are a good example of how to take advantage of the error fail-safe feature built into WordPerfect. But you may have an occasion someday to construct a macro that ignores error conditions, or even does some specific action when an error occurs. Wordperfect 5.1 permits both.

Conditional Chains

WordPerfect 4.2 provided a method of creating semi- intelligent macros by the use of "conditional chains." The technique takes advantage of the way the program's search feature works.

WordPerfect 5.1 does not allow you to construct conditional chains in the same manner, requiring instead that you write a conditional macro using programming statements, as described in Chapter 4. If you have version 4.2 conditional macros, you'll need to rewrite them in order to perform the same function in version 5.1.

CHAPTER 3
EDITING MACROS

If you don't like the way a macro works after you create it, don't start over -- edit it. WordPerfect 5.1 has its own built- in editor where you can modify any macro. You can even create macros from scratch using the editor.

You'll learn the basics of editing macros in this chapter. You'll also discover how to reassign nearly any key on your computer to another function and how to create chained and nested macro while using the macro editor. The techniques you encounter in this chapter are used in the remainder of this book, and form the backbone of WordPerfect <u>power macros</u>.

ACCESSING THE MACRO EDITOR

The macro editor can be access in two ways:

1. Edit an existing macro. Press **[Ctrl]**-**[F10]**. Type the name of the macro, and press **[Enter]**. Press **E** at the prompt to edit the macro.
2. Create a new macro from scratch. Press **[Home]**, then press **[Ctrl]**-**[F10]**. Type the name of the new macro, and press **[Enter]**. Provide a description (optional).

The macro editing window now appears. The contents of the macro is shown in the box. If you're editing a "dummy" macro (macro defined without any keystrokes), you'll see just a {DISPLAY OFF} code in the macro editor. If you're editing a macro that you've defined with keystrokes, you'll see all the keys you pressed.

To leave the editing window, press the **[F7]** to Exit. Upon exiting, the macro is automatically saved. If you want to leave the macro editor <u>without</u> saving any changes:

	Key Sequence	**What it Does**
1.	**[F1]**	Chooses Cancel key.
2.	**Y**	Answers Y, you do want to leave the editor without saving changes.

Macros that you've previously recorded appear with all the keystrokes you entered. Codes shown in boldface and in braces represent WordPerfect function or editing keys, such as **{Screen}** or **{Bold}**. In the editor, these function and editing codes are considered one character. You enter the codes by pressing the relevant key, and delete the entire code by pressing the **[Backspace]** or **[Delete]** key once. Alphabetic, numeric, and symbol keys appear in the macro editor window as single, unbolded characters.

Moving Within the Macro Editor

You move the cursor around within the macro editor box the same way as you do within WordPerfect's main document screen. The following table provides a quick run-down of the cursor movement keys, and what they do.

Table -- Cursor Movement Within Macro Editor

To	Press

Move to the beginning of the macro	**[Home]**, **[Home]**, **[Up]**
Move to the end of the macro	**[Home]**, **[Home]**, **[Down]**
Move to the top top of the box	**[Home]**, **[Up]**
Move to the bottom of the box	**[Home]**, **[Down]**
Move down one screen-full at a time	**[Page Down]**
Move up one screen-full at a time	**[Page Up]**
Move to beginning of current line	**[Home]**, **[Left]**
Move to end of current line	**[End]** or **[Home]**, **[Right]**
Move right one word	**[Ctrl]**-**[Right]**
Move left one word	**[Ctrl]-[Left]**

The **[Up]**, **[Down]**, **[Right]**, and **[Left]** cursor keys function as usual.

Note that some of the standard WordPerfect cursor movements keys are not active within the macro editor. These are:

. Escape -- to move the cursor x number of spaces, lines, or whatever.

. Goto -- to go to a different page.

Using Function and Editing Keys

While working inside the macro editor, the standard editing keys -- such as **[Backspace]** and **[Delete]** -- function as they normally do in WordPerfect. The next table lists the editing keys you can use with the macro editor, and what they do.

Table -- Function and Editing Keys in Macro Editor

To	Press
Delete one character/code to left of cursor	**[Backspace]**
Delete one character/code at the cursor	**[Delete]**
Delete all chars/codes to the end of line	**[Ctrl]-[End]**
Delete word at cursor	**[Ctrl]-[Backspace]**

Standard WordPerfect editing keys you can't use in the macro editor are:

. Delete to End of Page -- to delete all characters and codes from the cursor to the end of the macro document or screen.

. Home, Backspace -- to delete all characters of the current word to the left of the cursor.

. Home, Delete -- to delete all characters of the current word to the right of the cursor.

Inserting Editing Keys as Codes

If you want to place the definitions -- or codes -- of editing and cursor movement keys in the macro, do one of the following:

. Press **[Ctrl]**-**V**, then immediately press the editing key you want included in the macro. After that, the editing keys revert to their normal operation. Example: **[Ctrl]**-**V** **[Left]** places a **{Left}** code in the macro.

. Press **[Ctrl]**-**[F10]** to turn editing off. Now, <u>every</u> key you press is included in the macro. Press **[Ctrl]**-**[F10]** once more to turn editing back on.

You use the **[Ctrl]-V** or **[Ctrl]**-**[F10]** keys to turn editing on and off for additional keys, as well. The following table lists those keys that must be used with either **[Ctrl]-V** or **[Ctrl]**-**[F10]** so the appropriate code can be entered into the editor.

Table -- Keys that Need **[Ctrl]**-V to Add Key Code

Function	Key
Cancel	**[F1]**
Exit	**[F7]**
Tab	**[Tab]**
Enter	**[Enter]**
Right cursor	**[Right]**
Left cursor	**[Left]**
Up cursor	**[Up]**
Down cursor	**[Down]**
Home	**[Home]**
Typeover	**[Insert]**
Delete	**[Delete]**
Backspace	**[Backspace]**
Page Up	**[Page Up]**
Page Down	**[Page Down]**
Screen Up	**-** (on keypad)
Screen Down	**+** (on keypad)
End	**[End]**
Delete to end-of-line	**[Ctrl]-[End]**
Delete Word	**[Ctrl]**-**[Backspace]**
Word Right	**[Ctrl]**-**[Right]**
Word Left	**[Ctrl]**-**[Left]**
Escape	**[Esc]**
Macro Commands	**[Ctrl]**-**[Page Up]**
Help	**[F3]**

The **[Ctrl]**-**V** sequence is used for entering two other types of macro codes, as well. These are:

- Numbered variables, such as {VAR x} (where "x" is a number from 0 to 9).
- Nested Alt-key macros, such as {ALTx} (where "x" is a letter from A to Z).

The function of these codes is explained in Chapter 4, "Learning the Macro Programming Language."

Tip: If you press a key and WordPerfect responds differently, a modified keyboard layout may be currently in use. A keyboard layout redefines one or more of the keys on the keyboard. You might switch the functions of the Help and Cancel keys, for example, or modify the **[Backspace]** key so that it behaves differently than standard WordPerfect operation. In these instances, pressing a key may have a wildly different effect than you're expecting. If you think a keyboard layout is is the cause of your troubles, at the WordPerfect main editing screen choose the Setup key (**[Shift]**-**[F1]**), and note the entry beside the Keyboard Layout option. The currently active keyboard layout, if any, will be shown. You can quickly cancel any layout that's currently active by pressing **[Ctrl]**-**6**.

Programming Codes

The macros you edit will most likely begin with a **{DISPLAY OFF}** code. If, when you defined the macro from the keyboard, you've turned the display off or on using the technique described in Chapter 2, "Adding Power to Macros," you may see additional **{DISPLAY ON}** and **{DISPLAY OFF}** codes. Too, pauses that you entered during macro definition will appear as **{PAUSE}**.

Codes that appear in all upper-case and within braces are WordPerfect's programming commands. **{DISPLAY ON}**, **{DISPLAY OFF}**, and **{PAUSE}** are just three of among 56 commands you can insert into a WordPerfect macro. While we'll discuss the macro programming language in detail in subsequent chapters, you should know how these commands are added from within the macro editing window.

To insert a macro command, press the **[Ctrl]**-**[Page Up]** keys. A pop-up window appears over the macro editor box. You then use the cursor keys to select the command you want to insert, and press the **[Enter]** key when you're done.

Formatting the Macro Code

Note that pressing the **[Tab]** key tabs a line of macro code over approximately five spaces (two spaces at the beginning of the line), and that pressing the **[Enter]** key ends a line. The visual effect of these keys in the macro editing box does not relate to the operation of the macro. Adding four or five blank lines between segments of macro code does <u>not</u> produce four or five blank lines when the macro is executed.

You can "format" your macros for readability by adding Tabs and Enters as desired. The macro is now easier to read because you've separated its functions into logical units. When this macro is executed, the Tabs and Enters <u>will not</u> be included in the document.

Tip: Get into the habit of formatting your macros, especially if you plan to edit them or add additional program codes (as explained in Chapter 4). Develop a consistent style for macro formatting. Although adding Tabs and Enters for formatting purposes does increase the size of the macro by a small degree, the visual appeal and easy-of- use are well worth it.

To add **[Tab]** and **[Enter]** codes that are applied to the document during execution, enter them explicitly by first pressing either **[Ctrl]**-**V** or **[Ctrl]**-**[F10]**. When this macro is run, the Tabs and Enters <u>will</u> be included in the document.

Adding Comments to Macros

A technique often used by programmers is adding explanations or <u>comments</u> within their programmed code. You can do the same with WordPerfect macros. These comments don't execute when the macro is run, but they appear in the macro editor as a means to help you describe the workings of the macro, or to help you remember how you constructed a given macro.

To add a comment, press the **[Ctrl]**-**[Page Up]** key. If it isn't already, highlight the **{;} Comment** code, and press **[Enter]**. This inserts the **{;}** comment code in the macro. Write your comment, up to any length, and end it with a tilde (~) character. The tilde marks the end of the comment, and tells WordPerfect it that everything after this mark is to be executed.

> **Tip**: You may wish to include comments when you are defining a macro from the keyboard. You do so using the same basic techniques as turning the display off and on or adding a pause, except that you choose the Comment option to insert a comment. Write the comment and press the **[Enter]** key when you're done.

Macro Editor Features

Other than adding and deleting pieces of macro code, the editor is very crude and lacks copy and paste editing features. The macro editor does not allow you to copy a segment of code from one part of the box and paste it into another. If you find you need to move a segment of code to another area of the macro, make a note of it and manually retype it. Double check that you have re-entered the code <u>exactly</u> as the original, then erase the original.

These capabilities <u>are</u> provided by EDitor 3.0, available separately from WordPerfect Corp. The ED program, which is included in the WordPerfect Office package, lets you edit macros created by all WordPerfect Corp. products.

You can, however, include the text characters of a **[Alt]**-key macro by running the macro during editing. For example, suppose you are editing a macro for preparing a boilerplate customer reply letter. You have a large paragraph that you don't want to retype. Position the cursor in the editing box where you want the text to appear, then press **[Alt]** and the appropriate letter. The **[Alt]**-key macro executes, as normal, and its text is entered into the new customer reply macro.

Note that this technique only works with **[Alt]**-key macros. If the macro you want to use isn't an **[Alt]**-key macro, you can easily convert it simply by renaming it. You can use DOS or WordPerfect's List Files feature to rename a macro file.

> **IMPORTANT**: WordPerfect will act "dead" if you try to load the contents of an **[Alt]**-key macro when that macro starts with a {DISPLAY OFF} code (WordPerfect automatically adds this code to all new macros it creates, but never adds the {DISPLAY ON} code at the end). You need to edit the **[Alt]**-key macro <u>before</u> you use it and remove the {DISPLAY OFF} code. If you forget to do this, press **[F1]** to cancel the operation and WordPerfect will come back to life.

UNDERSTANDING THE [Ctrl]-[Page Up] KEY

The **[Ctrl]**-**[Page Up]** key is multi-faceted, and performs many duties, depending on what you're doing with WordPerfect at the time. There are four uses of **[Ctrl]**-**[Page Up]**, some of which we have already covered.

1. <u>From the WordPerfect document editing screen:</u> pressing **[Ctrl]**-**[Page**

Up] allows you to assign values to variables. Variables are temporary holders of information (such as text or numbers) that you can retrieve later or use in a macro. Variables are detailed more fully in future chapters.

2. When defining a macro: pressing **[Ctrl]**-**[Page Up]** lets you add a comment, add a pause, assign a value to a variable, or turn the display off and on.

3. When in the macro editor: pressing **[Ctrl]**-**[Page Up]** accesses the macro programming commands.

4. When in the macro editor: pressing **[Ctrl]**-V, then **[Ctrl]**-**[Page Up]** inserts a special {Macro Commands} code in the macro. This code has special use when working with variables or when creating macros that actually write other macros.

MACRO EDITOR CODES

The macro editing window displays the meaning of the WordPerfect's function and editing keys as codes. Most codes are self-explanatory, but if you find a code that is new to you, look it up in the table that follows.

Table -- Macro Editor Codes and Corresponding Keys

Code	Press	Comments
{-}	Alt--	Alt and minus key
{ }		Special access; see Chapter 19
{^\}	Ctrl-\	
{^_}		Special access; see Chapter 19
{^A}	Ctrl-A	
{^B}	Ctrl-B	
{^C}	Ctrl-C	
{^D}	Ctrl-D	
{^E}	Ctrl-E	
{^F}	Ctrl-F	
{^G}	Ctrl-G	
{^M}	Ctrl-M	
{^N}	Ctrl-N	
{^O}	Ctrl-O	
{^P}	Ctrl-P	
{^Q}	Ctrl-Q	
{^R}	Ctrl-R	
{^S}	Ctrl-S	
{^T}	Ctrl-T	
{^U}	Ctrl-U	
{^V}	Ctrl-V, Ctrl-V	
{^]}	Ctrl-]	
{Alt-Home}	Alt-Home	
{Backspace}	Ctrl-V, Backspace	
{Block Copy}	*Ctrl-Ins	From macro command list
{Block Move}	*Ctrl-Del	From macro command list
{Block Append}		From macro command list
{Block}	Alt-F4	

{Bold}	F6	
{Cancel}	Ctrl-V, F1	
{Center}	Shift-F6	
{Columns/Tables}	Alt-F7	
{Compose}		Special access; see Chapter 19
{Date/Outline}	Shift-F5	
{Del Word (Row)}		Special access; see Chapter 19
{Del to EOP}	Ctrl-V, Ctrl-Page Down (Also Ctrl-L)	
{Del to EOL}	Ctrl-V, Ctrl-End (Also Ctrl-V, Ctrl-K)	
{Delete Word}	Ctrl-V, Ctrl-Backspace	
{Delete}	Ctrl-V, Delete	
{Down}	Ctrl-V, Down (Also Ctrl-Z)	
{End Field}	F9	
{End}	Ctrl-V, End	
{Enter}	Ctrl-V, Enter (Also Ctrl-V, Ctrl-J)	
{Escape}	Ctrl-V, Esc (Also Ctrl-V, Ctrl-[)	
{Exit}	Ctrl-V, F7	
{Flush Right}	Alt-F6	
{Font}	Ctrl-F8	
{Footnote}	Ctrl-F7	
{Format}	Shift-F8	
{Goto}	Ctrl-V, Ctrl-Home	
{Graphics}	Alt-F9	
{Hard Page}	Ctrl-Enter	
{Help}	F3	
{Home-Home-Left}		Special access; see Chapter 19
{Home}	Ctrl-H	
{Indent}	F4	
{Invalid}		Special access; see Chapter 19
{Item Left}	*Alt-Left	From macro commands list
{Item Right}	*Alt-Right	From macro commands list
{Item Up}	*Alt-Up	From macro commands list
{Item Down}	*Alt-Down	From macro commands lis
{Keyboard}		Special access; see Chapter 19
{L/R Indent}	Shift-F4	
{Left Margin Rel}	Shift-Tab	
{Left Search}	Shift-F2	
{Left}	Ctrl-V, Left (Also Ctrl-Y)	
{List Files}	F5	
{Macro Define}	Ctrl-V, Ctrl-F10	Code not active
{Macro}	Alt-F10	
{Mark Text}	Alt-F5	
{Menu Bar}	Alt-=	Alt and equals key
{Merge Codes}	Shift-F9	
{Merge/Sort}	Ctrl-F9	
{Move}	Ctrl-F4	

{Page Down}	Ctrl-V, Page Down	
{Page Up}	Ctrl-V, Page Up	
{Para Up}	*Ctrl-Up	From macro commands list
{Para Down}	*Ctrl-Down	From macro commands list
{Print}	Shift-F7	
{Replace}	Alt-F2	
{Retrieve}	Shift-F10	
{Reveal Codes}	Alt-F3	
{Right}	Ctrl-V, Right (Also Ctrl-X)	
{Save}	F10	
{Screen Down}	Ctrl-V, Screen Down	
{Screen Up}	Ctrl-V, Screen Up	
{Screen}	Ctrl-F3	
{Search}	F2	
{Setup}	Shift-F1	
{Shell}	Ctrl-F1	
{SHy}	Ctrl--	Ctrl and minus key
{Spell}	Ctrl-F2	
{Style}	Alt-F8	
{Switch}	Shift-F3	
{Tab Align}	Ctrl-F6	
{Tab}	Ctrl-V, Tab (Also Ctrl-V, Ctrl-I)	
{Text In/Out}	Ctrl-F5	
{Thesaurus}	Alt-F1	
{Typeover}	Ctrl-V, Insert	
{Underline}	F8	
{Up}	Ctrl-V, Up (Also Ctrl-W)	
{Word Left}	Ctrl-V, Ctrl-Left	
{Word Right}	Ctrl-V, Ctrl-Right	

Additional Codes

Code	Meaning
{COMMAND}	Macro command code (from commands list)
{ALT x}	Nested Alt-key macro
{VAR x}	Numbered variable
{KEY MACRO X}	Nested keyboard macro

Note: the "x" in the above codes represent a letter or number, and that {COMMAND} is representative of the commands in the macro commands list.

EDITING THE MACRO DESCRIPTION

The one-line description provides a quick-reference to the purpose of the macro. You don't have to describe the macro if you don't want to when you first create the macro, and you always have the chance to edit the description later.

	Key Sequence	**What it Does**
1.	**[Ctrl]**-**[F10]**	Starts macro definition.

2. {<u>name</u>} **[Enter]**	Enters the name of the existing macro.
3. **3** or **D**	Chooses the edit Description option.

You may now edit the description as you like. The editing keys function as they normally do in WordPerfect. With the cursor positioned at the first character, typing anything erases the entire line. Press the **[Enter]** key when you are done.

CHAPTER 4
LEARNING THE MACRO PROGRAMMING LANGUAGE

Being able to record a series of keystrokes for later recall is power enough, but WordPerfect's macro feature also includes a method of programming your macros so that they behave like stand- alone applications.

With just a little bit of ingenuity, you can create macros that create, from scratch, a complete invoice for services or products, an electronic while-you-were-out telephone note pad, a note pad, mini calculator, a self-testing and scoring quiz game, a text adventure game, and much, much more. In short, there is almost no limit to what you an do with WordPerfect's macro programming language.

This chapter explains how how to access the programming feature and the meaning of the various programming commands or instructions. This chapter does not delve too deeply in actually employing the various programming commands in real macros -- the rest of the book is devoted to that. If you are serious about macro programming, be sure to also read Part Three of this book. The chapters in this section provide additional insights on applying the macro programming language, as well as how to avoid its frustrating pitfalls.

THINKING LIKE A PROGRAMMER

Although you don't need any previous training or experience at computer programming to master WordPerfect's programming language, it does help. At the very least, you should learn how to think like a programmer.

It is not the intent of this book to teach programming principles; that would require far too much space. However, you can gain a good grasp of programming fundamentals by concentrating on 10 main areas of macro design:

. Flow Control
. Routines
. Subroutines
. Variables
. Expressions
. Strings
. Conditional Statements
. Branching
. Looping
. Entering Data
. Outputting Data

Let's take a closer look at each.

Going With the Flow

Simple, one-function macros can be created without a blueprint or <u>flow chart</u> but you should give extra thought and consideration to macros that are more complex in nature. You may find it helpful to draw a programming flow chart that includes the basic steps of the macro. In each box is a complete step; arrows connect the boxes to indicate the progress of steps throughout the macro. Flow charts are particularly handy when creating chained and nested macros, or macros that consists of many self-contained routines, because the drawing helps you visualize the function and flow of each macro.

While you are planning the macro, note the steps required to create it. On a scrap document, go through the motions of executing the macro -- but don't record it just yet. On a piece of paper, note the commands and menu options you choose to complete the task. Don't forget the **[Enter]** key, as well as all other keys you must press to make the macro work properly. Be sure to note each time you press it. When you think you have the procedure mapped out, define the macro and record it.

If you make a mistake, you can either cancel the recording by pressing **[Ctrl]**-**[F10]** again or edit out the errors later. Small mistakes usually make no difference, but you'll want to get into the habit of cleaning up your macros once you have created them.

After defining the macro, try it out to make sure it works the way you originally planned. If you spot a mistake, double- check the steps you took and redefine the macro.

The Benefit of Routines

One of the first things a programmer does when he or she starts on a project is to map out the individual segments, or <u>routines</u>, that make up the software. Even the longest, most complex program -- and this includes macros -- consist of little more than bite-size routines. The macro progresses from one routine to the next in an orderly and logical fashion.

What's a routine? A routine is any self-contained segment of code that performs a basic action. In the context of macros, a routine can be a single command or character of text, or it can take up the majority of a mammoth 20K macro. Suppose your macro resets the active printer, formats the document to legal size, and prints it. The macro can be divided into three distinct routines:

. Routine 1: Reset the active printer.
. Routine 2: Format the document to legal size.
. Routine 3: Print the document.

While there's no need to physically separate these routines within a macro, or even create separate macros and chain to each one in turn (although this <u>can</u> be done, as explained in Chapter 2), it helps to think of the macro as being composed of these more basic parts. If the macro doesn't work properly, you can more easily analyze the problem if you can narrow it down to a specific segment of the code. For example, if the macro is resetting the active printer and printing the document, but not properly formatting the document to legal size, you can readily trace the problem to the number 2 routine.

Many macros simply start at the beginning and advance one step at a time to the end, taking each instruction in turn and acting upon it. This is the approach of our example macro just cited.

But some WordPerfect macros jump around. They may start with routine 1, but then jump to routine 3, back to 1, then move on to 2. Jumping around within the macro pays off when you need to re-use a routine two or more times. Instead of repeating the steps of the routine each time, you can simply direct the macro to repeat the same routine.

Here's another benefit of working with discrete routines. Depending on the application of the macro, it may behave one way when used with a particular type of document, and behave in a totally different way with another document. This concept may take some time to get used to, but it lends a great deal of power and flexibility to your macros. Instead of your macros conforming to a strict procedure, they are free to interact and adapt with your documents. In other words, your macros are "intelligent," automatically adapting to the situation and document at hand.

WordPerfect macros don't use line numbers, as found on some other programming languages, such as BASIC. You identify routines by name, or label. When you instruct WordPerfect to jump from one routine to another, you tell it which label to go to. Of course, if your macro is designed to go from beginning to end without stopping, you don't need labels, but it's still a good idea to think of the construction of the macro as discrete routines.

Subroutines

Subroutines are similar to routines, but are specifically designed to be used by the macro as a sub-component. Most often, subroutines are activated only when certain criteria exists -- an error occurs or a search fails, for instance. These are routines that the macro branches off to temporarily, in order to accomplish some task or to wait until a certain condition is met. When the subroutine ends, the macro jumps back to where it left off.

Like a routine, a subroutine consists of a label (so the macro can find it), but it also has a return identifier. The return marks the end of the subroutine, and instructs the macro to go back to where it first left off.

Variables

A variable is a special holding tank for information, sort of a placeholder. WordPerfect has three types of variables, numbered, named, and internal.

Numbered variables, a hold-over from version 5.0 of WordPerfect, are identified by a single-digit number, from 0 to 9. You can access these numbered variables at any spot within WordPerfect by pressing a number with the **[Alt]** key. For instance, suppose you've previously assigned the text

Hello there

to variable 4. Press **[Alt]**-**4** and WordPerfect types "Hello there" at the cursor.

You'll have plenty of cause to use variables in more sophisticated programmed macros, but you can also use the feature to create up to 10 more temporary macros (in addition to the one that you can create with the **[Enter]** key). The limitations of variables is that they can contain numbers and text only, making them best suited for adding boilerplate text. Unless you use more sophisticated programming, you can't code them with commands, cursor movements, or other keyboard functions.

WordPerfect macros also support named variables. These are variables you assign within the macro editor, and are used to "remember" a piece of vital information so that it can be used elsewhere in the macro, or even in another macro.

Internal variables are variables generated by WordPerfect and reflect certain operating characteristics or states. For example, one internal variable keeps track of the current page number; another contains the absolute position of the cursor on the page.

Expressions

The term expressions, as used in WordPerfect macros, is a programmer's word that means pretty much what it means to the rest of the world. You'll be seeing the word many times throughout this book, so it's a good idea to learn how it relates to macro programming.

An expression is a procedure a WordPerfect macro must follow to complete a task you've given it. Sometimes the expression is a simple variable -- "take the contents of this variable and apply it to that command." Often times, the expression is a little more complex, like take the number already in a variable, and add another one to it.

When you ask a WordPerfect macro to perform some type of calculation or thinking process, you're asking it to evaluate the expression. For example, if the expression reads "1+1," then the macro must first evaluate it (add one and one to make two) before proceeding.

Some more advanced macro expressions may appear exotic, but you'll have plenty of chance to use them. The most common is the evaluating if a statement is TRUE or FALSE (the words TRUE or FALSE are capitalized, to show you we're dealing with logical functions). Here's a good example of a TRUE/FALSE expression that must be evaluated by the macro:

```
IF variable 1 = 10 THEN QUIT
```

In reading the expression (which isn't in acceptable macro notation, by the way, but it's close), it reads "IF the contents of variable 1 is equal to 10, then stop." Before proceeding with the remainder of the macro, WordPerfect must pause, take a peek inside variable 1, and apply it to the logical equation. If the result is TRUE, then the macro ends. If it's FALSE (variable 1 has a number other than 10), then something else happens.

Strings

A common term in programming circles is the string. A string is simply a sequence of alphabet or number keys. In the context of macros, strings are used in variables. Once stored in a variable, the string can be acted upon by the macro.

For example, if the string is a number, you can multiply it with another number or the numeric contents of another variable. If the string is text, you can compare it with another string to see if the two are the same. The concept of comparing string comes in handy when designing interactive macros. The advanced macros in Part Two of this book rely heavily on comparing strings.

Conditional Statements

You've already learned that macros can be constructed so that they perform certain routines in one instance, and other routines in another. The macro is responding to specific conditions, set either by the user, by the document, by information in the document, or by some other variable.

A <u>conditional statement</u> is a fork in the road with a choice of two directions to take, depending on the response to a simple true/false question. WordPerfect macros provide many ways to create conditional statements, but they all have one thing in common: to activate a certain routine (or path of routines) depending on external data.

Here's an example of a conditional statement. "If it's cold outside, I'll wear my jacket. Otherwise, I'll leave the jacket at home." The statement can be broken down into three segments:

. The condition to be met (if it's cold).
. The result if the condition is TRUE (wear the jacket).
. The result if the condition is FALSE (leave the jacket at home).

Obviously, this isn't the kind of conditional statement you'll write with WordPerfect macros. But you may design a conditional statement that starts one macro depending on one type of user input, or another macro depending on other types of user input.

All conditional statements have a condition that must be met, and a specified action if the condition is TRUE or FALSE. Not all conditional statements specify an action for both a TRUE or FALSE, but most do.

Branching

Akin to the conditional statement is the branch, where the macro has two or more paths to take depending on external criteria. A good example of a branch is a menu selection. Here, you have four choices:

. Edit a Document.
. Print a Document.
. Format a Document.
. Quit WordPerfect

Each of these choices is a branch. Each branch specifies one or more unique routines in the macro. The Edit a Document branch, for example, will not use the same routines as Print a Document, Format a Document, or Quit WordPerfect.

Looping

A <u>loop</u> is a routine that is repeating two or more times. A typical loop is a "keyboard scan" where the macro constantly checks the keys you press. If you press the key the loop routine is looking for, the macro breaks out of the loop and continues on with some other routines. If the key isn't the one that the loop is looking for, the routine is restarted again.

Loops are really specialized versions of conditional statements. Instead of stopping to provide a choice of two directions, the macro is designed to continue the loop until a certain condition is met. WordPerfect macros contain many sophisticated ways to construct loops. You'll find loops helpful in tackling the most demanding macro assignments.

Entering Data

The basic WordPerfect macro is merely a recording of a series of keystrokes. But you can also program macros to stop and wait for user input. Once the user has entered the data, the macro uses the information -- be it text, a number, or a single keypress -- to complete its task.

You might pause a macro to allow the user to enter her name, the current date, or some other variable information. Your macro may then use that information to create a mailing label, or to date a series of database entries. Input data can be entered directly into the document, or can be temporarily stored in a variable. The macro can then sample the contents of the variable, and use it in a conditional statement, a branch, or a loop -- or even save it for use later on.

Outputting Data

WordPerfect macros output data in two basic ways:

. Characters entered into the document. The macro "types" text for you, as when creating a form letter using boilerplate text recorded previously in a macro.

. Replay of function and editing keys. The macro actuates one or more of WordPerfect's function and editing keys, thereby automatically controlling its operating. The macro may reset tabs and margins for example, using the options under the Format key.

ACCESSING MACRO COMMANDS

The commands in WordPerfect's macro programming language is available while in the macro editing window. To access the programming statements while editing a macro, press **[Ctrl]**-**[Page Up]**. A pop-up menu appears over the editing box. Use the up and down arrow keys to scroll through the contents of the programming menu. You can quickly shuttle to a particular statement by typing the first couple of characters in its name.

Each statement, such as {DISPLAY ON} and {CHAIN}, is enclosed in brackets. Some statements are followed by one or more arguments. These arguments tell WordPerfect what to do with the statement. For example, the {CHAIN} statement is followed by a macro file name argument. To complete the statement in an actual macro, you'd enter:

{CHAIN}filename~

WordPerfect uses tilde character as a delimiter. The program knows that the argument is done when it reaches the tilde, and that it should expect text or another programming code to follow.

To enter a programming command (otherwise known as a statement; we'll use the two words interchangeably throughout the book) in the macro editing box, select it and press **[Enter]**. Note that the text for the argument is not included, only the command itself, and that the command is shown in bold.

You may exit the pop-up menu without making a selection by pressing the **[Esc]** key. If you press **[Ctrl]**-**[Page Up]** or **[Esc]** again, WordPerfect reselects the macro command that you last used. This allows you to quickly enter a series of the same statements in the macro editing box.

MACRO COMMAND TYPES

It's helpful to categorize WordPerfect's macro commands into discrete types. For consistency, we'll retain the categorization that WordPerfect uses in its documentation (and we've added one of our own).

There are nine categories of macro commands:

. User interface
. Flow control
. Macro and subroutine termination
. External condition handling
. Macro execution
. Macro execution control
. Variables
. Programming aids
. Special Purpose

Note that some commands serve double duty. For example, the {DISPLAY OFF} command is found in both the macro execution control and programming aides categories.

User Interface Commands

The user interface commands allow input from the user and/or display a message on the screen. The user interface commands are:

{BELL} -- Sounds a short warning tone.
{CHAR} -- Allows for single character user input.
{INPUT} -- Allows for multiple character user input.
{LOOK} -- Samples the key you just pressed.
{PAUSE} -- Pauses macro execution and allows for keyboard entry.
{PROMPT} -- Displays a message on the screen.
{STATUS PROMPT} -- Displays a semi-permanent message on the screen.
{TEXT} -- Allows for multiple character user input.

Flow Control Commands

Flow control commands redirect macro execution depending on external criteria (such as user input). The flow control commands are:

{BREAK} -- Prematurely terminates a conditional statement, loop, or macro.
{CALL} -- Calls a subroutine.
{CASE} -- Establishes one or more branches.
{CASE CALL} -- Establishes one or more branches, with each branch executed as a subroutine.
{CHAIN} -- Executes another macro.

{ELSE} -- Establishes a FALSE condition for an IF statement. {END FOR} -- Ends a FOR and FOR EACH loop.
{END IF} -- Ends an IF statement.
{END WHILE} -- Ends a WHILE loop.
{FOR} -- Establishes a self-contained "counter" loop.
{FOR EACH} -- Applies one or more commands or routines to a single variable.
{GO} -- Jumps to a labeled routine.
{IF} -- Establishes an IF conditional statement and what happens when the statement is TRUE.
{IF EXISTS} -- Checks if a variable is empty (doesn't exist).
{LABEL} -- Names a routine.
{NEST} -- Temporarily executes another macro, then returns to the original macro.
{NEXT} -- Increments the counter in a FOR, FOR EACH, and WHILE loop.
{ON CANCEL} -- Provides an alternative response in case the macro is canceled.
{ON ERROR} -- Provides an alternative response in case an error occurs during macro execution.
{ON NOT FOUND} -- Provides an alternative response in case a search fails during macro execution.
{QUIT} -- Stops the macro.
{RESTART} -- Stops all macro execution at the end of a nested macro.
{RETURN} -- Returns from a nested macro or a subroutine.
{RETURN CANCEL} -- Generates a "cancel condition."
{RETURN ERROR} -- Generates an "error condition."
{RETURN NOT FOUND} -- Generates a "not found condition."
{SHELL MACRO} -- Starts a shell macro.
{WHILE} -- Runs a self-repeating loop until a condition is met.

Macro and Subroutine Termination Commands

The macro and subroutine termination commands end macro execution, or break out of subroutines and continues with the rest of the macro. The macro and subroutine termination commands are:

{BREAK} -- Prematurely terminates a conditional statement or macro.
{QUIT} -- Stops the macro.
{RESTART} -- Stops all macro execution at the end of a nested macro.
{RETURN} -- Returns from a nested macro or a subroutine.
{RETURN CANCEL} -- Generates a "cancel condition."
{RETURN ERROR} -- Generates an "error condition."
{RETURN NOT FOUND} -- Generates a "not found condition"

External Condition Handling Commands

The external condition handling commands specify how a condition that occurs outside the macro is handled. In the case of the RETURN commands, they generate or create the condition.   The external condition handling commands are:

{CANCEL OFF} -- Disables the Cancel key.
{CANCEL ON} -- Enables the Cancel key.
{ON CANCEL} -- Provides an alternative response in case the macro is canceled.
{ON ERROR} -- Provides an alternative response in case an error occurs during macro execution.
{ON NOT FOUND} -- Provides an alternative response in case a search fails during macro execution.
{RETURN CANCEL} -- Generates a "cancel condition."
{RETURN ERROR} -- Generates an "error condition."
{RETURN NOT FOUND} -- Generates a "not found condition."

Macro Execution Commands

The macro execution commands run a macro.
{CHAIN} -- Starts a new macro.
{NEST} -- Temporarily executes another macro, then returns to the original macro.
{SHELL MACRO} -- Starts a Shell macro.

Macro Execution Control Commands

Macro execution commands affect the operation of macros by either controlling the display, macro execution speed, or delay until the next command is executed. The macro execution control commands are:

{DISPLAY OFF} -- Turns the display off.
{DISPLAY ON} -- Turns the display on.
{MENU OFF} -- Turns menu display off.
{MENU ON} -- Turns menu display on.
{SPEED} -- Slows down macro execution.
{WAIT} -- Waits a predetermined time before continuing.

Variable Commands

The variable commands assign values to variables or determines some characteristic or state of a variable. The variable commands are:

{ASSIGN} -- Assigns a value to variable.
{CHAR} -- Displays a message on the screen and assigns the next keypress to a variable.
{IF EXISTS} -- Checks if a variable is empty.
{LEN} -- Counts the number of characters in a variable.
{LOOK} -- Assigns the next keypress to a variable.
{MID} -- Captures a sub-string of a variable.
{SYSTEM} -- Contains any of several internal WordPerfect variables.
{TEXT} -- Displays a message on the screen and assigns one or more keypresses to a variable.
{VARIABLE} -- Displays the contents of variable.

Programming Aid Commands

Programming aid commands help make programming and "debugging" macros easier. The programming aid commands are:

{;} -- Provides non-executing comment within the macro.
{BELL} -- Sounds a short tone (audible feedback).
{SPEED} -- Slows down macro execution.
{STEP OFF} -- Disables {STEP OFF}.
{STEP ON} -- Executes one character or command at a time.

Special Purpose Commands

Some WordPerfect Macro commands defy easy categorization. These extend the usefulness of the macro language and are most often used with other commands, like {ASSIGN} or {IF}.

{KTON} -- Translate a key (alphabetic, numeric, symbol, or WordPerfect function) to its numbered equivalent.
{NTOK} -- Translates a number to its key equivalent.
{ORIGINAL KEY} -- Echos the last key pressed.
{STATE} -- Indicates one or more current operating states of WordPerfect.

Additional Macro Codes

Some useful macro codes are not found in the macro command list. These are:
{ALTx} -- Starts an Alt-key macro ("x" is a letter of the alphabet).
{KEY MACRO x} -- Runs a keyboard layout macro. The "x" is a number corresponding a key macro in the currently active keyboard layout.
{VAR x} -- Displays the contents of a variable. The "x" is a number from 0 to 9.

FOR BASIC PROGRAMMERS: COMPARING WORDPERFECT COMMANDS

If you're familiar with BASIC you'll find many of WordPerfect's macro commands are similar in function and syntax. A comparison of the functions may help you to understand how WordPerfect macro commands work.

Macro Command	**BASIC Keyword or Code**
{;}	REM
{ASSIGN}	LET
{BELL}	BEEP
{BREAK}	EXIT, EXIT FOR
{CALL}	CALL, GOSUB
{CANCEL OFF}	KEY OFF (**[F1]** key only)
{CANCEL ON}	KEY ON (**[F1]** key only)
{CASE}	SELECT CASE, ON...GOTO
{CASE CALL}	ON...GOSUB
{CHAIN}	CHAIN, RUN
{CHAR}	INPUT (single character), INKEY$
{DISPLAY OFF}	CLS
{ELSE}	ELSE
{END FOR}	EXIT

{END IF}	END IF
{END WHILE}	WEND
{FOR}	FOR
{GO}	GOTO
{IF}	IF (THEN implied)
{IF EXISTS}	IF var=""
{KTON}	STR$
{LABEL}	LABEL:, LineNumber
{LEN}	LEN
{LOOK}	INKEY$
{MID}	MID$
{NEST}	RUN
{NEXT}	NEXT
{NTOK}	CHR$
{ON CANCEL}	ON KEY (3B) (ON Cancel key event trap)
{ON ERROR}	ON ERROR
{PAUSE}	INPUT (no message)
{QUIT}	STOP
{RETURN}	RETURN
{RETURN ERROR}	ERROR
{STATE}	PEEK (specific internal variables only)
{STEP OFF}	TROFF
{STEP ON}	TRON
{SYSTEM}	ENVIRON$, PEEK (specific internal variables only)
{TEXT}	INPUT (multiple character)
{WAIT}	WHILE
{WHILE}	WAIT

ADDITIONAL MACRO CODES

The commands list in the pop-up macro editor window includes several other items that aren't really true "commands." These items, for moving the cursor to the next or previous paragraph, or to move the cursor within columns and tables, are duplicated with **[Ctrl]** and **[Alt]** keyboard equivalents. But these equivalents only work with enhanced keyboards (such as the 101- key keyboards found on the IBM PS/2 and similar models). If your keyboard can't generate these codes you can still access them in the macro editor.

. {Para Up} -- Moves the cursor to the preceding paragraph. Same as pressing **[Ctrl]**-**[Up]**.

. {Para Down} -- Moves the cursor to the next paragraph. Same as pressing **[Ctrl]-[Down]**.

. {Item Left} -- Moves the cursor to the column or table cell to the left. Same as pressing **[Alt]**-**[Left]**.

. {Item Right} -- Moves the cursor to the column or table cell to the right. Same as pressing **[Alt]**-**[Right]**.

. {Item Up} -- Moves the cursor to the column or table cell above. Same as pressing **[Alt]**-**[Up]**.

. {Item Down} -- Moves the cursor to the column or table cell below. Same as pressing **[Alt]**-**[Down]**.

Common keyboard codes found in macros are {Enter}, {Tab}, {ALTx}, and {VAR x}. These are not found in the pop-up commands list, but are entered directly from the keyboard. Here's what they do and how to enter them:

{Enter}

The {Enter} code within the macro indicates you want a hard return in the macro, or else want to simulate pressing the **[Enter]** key while in a menu, list, etc. Normally, pressing the **[Enter]** key inserts a blank line only. To include the {Enter} code in a macro, press **[Ctrl]**-**V** first, then press **[Enter]**.

{Tab}

The {Tab} key code within the macro indicates you want to tab some text to the next tab stop. Normally, pressing the **[Tab]** key inserts a empty spaces in a line, and is most often used to "format" the macro code for readability. To include the {Tab} code in a macro, press **[Ctrl]**-**V** first, then press **[Tab]**.

{ALTx}

The {ALT x} code ("x" is a letter of the alphabet) commands the macro to nest to an **[Alt]**-key macro. The {ALTx} code is the same as:

{NEST}altx~

when the {ALTx} code is not at the end of the macro.

{CHAIN}altx~

when the {ALTx} code is at the end of the macro.

To enter the {ALTx} code, first press **[Ctrl]**-**V**, then press the **[Alt]** key and a letter key. For example, to enter {ALTZ}, press **[Ctrl]**-**V**, **[Alt]**-**Z**.

{VAR x}

The {VAR x} code ("x" is a number from 0 to 9} indicates a numbered variable. You'll use the {VAR x} code to print the contents of a variable in a document, to assign one variable to another, and for many other tasks.

To enter the {VAR x} code, first press **[Ctrl]**-**V**, then press the **[Alt]** key and a number key (use the numbers along the top of the keyboard, not the ones in the numeric keypad). For example, to enter {VAR 1}, press **[Ctrl]**-**V**, **[Alt]**-**1**.