# Turbo Pascal 4.0 Supplement for

# Introduction to Pascal and Structured Design

Nell Dale
Chip Weems

Prepared by
Tom Parks

# TURBO 4.0 PASCAL SUPPLEMENT

for

# Introduction to Pascal and Structured Design

Second Edition


**Nell Dale**
University of Texas at Austin

**Chip Weems**
University of Massachusetts at Amherst

**Prepared by Tom Parks**





D. C. Heath and Company
Lexington, Massachusetts Toronto

Published simultaneously in Canada.

Printed in the United States of America.

International Standard Book Number: 0-669-20031-X

10 9 8 7 6 5 4 3 2

# Introduction

This supplement is meant to accompany *Introduction to Pascal and Structured Design*, Second Edition, by Nell Dale and Chip Weems. Used in conjunction with the text, it provides a useful reference for those students using a specific compiler in their course. Supplements are available for Turbo 3.0, Macintosh Pascal, and VAX Pascal.

This Turbo Pascal supplement has been coordinated with the order of presentation of topics in the text. Whenever an arrow and the symbol T (for Turbo) appear in the margin of the text, you should refer to the entry in the supplement. Each supplement entry will be marked by an arrow and the page number that corresponds to the text page.

# Contents

## Chapter 1 Overview of Programming

22→ This supplement describes the differences between Turbo Pascal version 4.0 (a product of Borland International, Inc.) and the International Standards Organization (ISO) standard for Pascal. It also describes many of the special features of Turbo Pascal. However, this supplement is *not* a complete manual for Turbo Pascal. Our intent is simply to provide you with enough information to be able to use the Turbo Pascal system in learning to program with Pascal. The Turbo Pascal Owner's Handbook provides a complete description of all of the features of the Turbo system.

We should also note that this supplement is based on Turbo Pascal Version 4.0 for the IBM PS/2 or PC and compatibles, running a PC-DOS or MS-DOS version 2.0 or higher. If you are using Turbo Pascal on some other system, you will have to consult your manuals for specific system commands. However, except where we've noted otherwise, the other information in this manual doesn't depend on a particular operating system.

You should now continue reading in the text on page 22.

25→ On the IBM PC, to "boot" the system, proceed as follows: If your PC has only floppy-disk drives, then insert the system disk in floppy-disk drive A (usually the left or top drive in a two-drive system). To insert the disk, lift the latch on the front of the floppy drive. Slip the disk out of its paper cover and orient it with the oval cutout toward the floppy drive and the label face up. Slide the disk all the way into the slot in the drive and close the latch. (If your PC has a hard disk, skip this step. In fact, for hard-disk systems, if there is a floppy disk in drive A, you should remove it.)

Next, turn on the PC (the switch is on the right side of the machine, near the back). If the PC is already turned on, you may boot it by pressing the Ctrl, Alt, and Del keys simultaneously.

Several messages will appear on the screen, indicating that the operating system has been loaded. You should then see the system prompt:

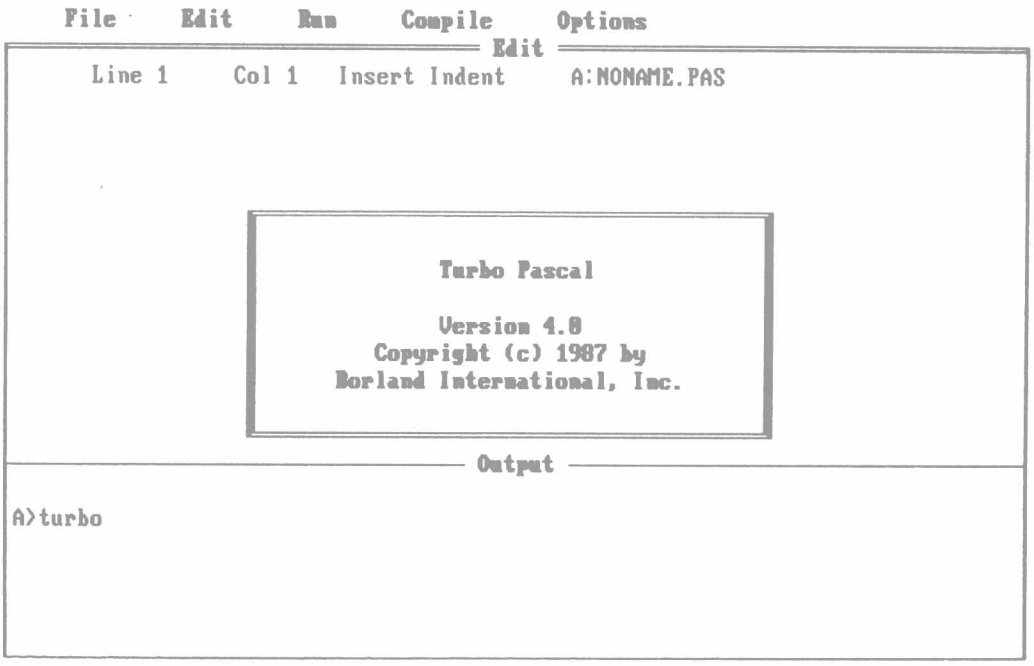
```
A>
```

Any time this prompt appears, you may type an operating system command. (On a hard-disk system, the prompt is C>.)

Now you are ready to start the Turbo Pascal system. (If Turbo Pascal is on a separate floppy disk, you will have to remove the system disk from drive A and insert the Turbo Pascal disk in its place.) Simply type Turbo, and press the return key to start Turbo Pascal (always press the return key when you are done typing a line on the keyboard). Note that there is also a command-line version of the compiler called TPC. The use of TPC requires less memory (256K as opposed to the 384K requirement of the Turbo command). To use

TPC, you will need to supply your own editor. This supplement will always refer to the integrated environment of the Turbo command.

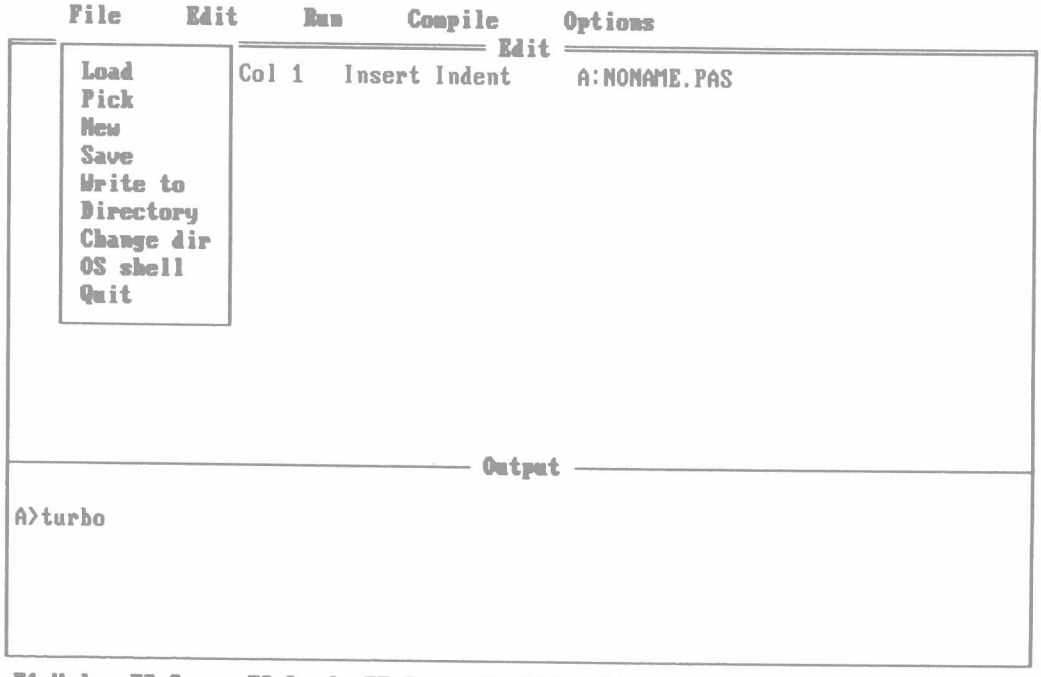
After you have entered the Turbo command, a menu appears on the screen:



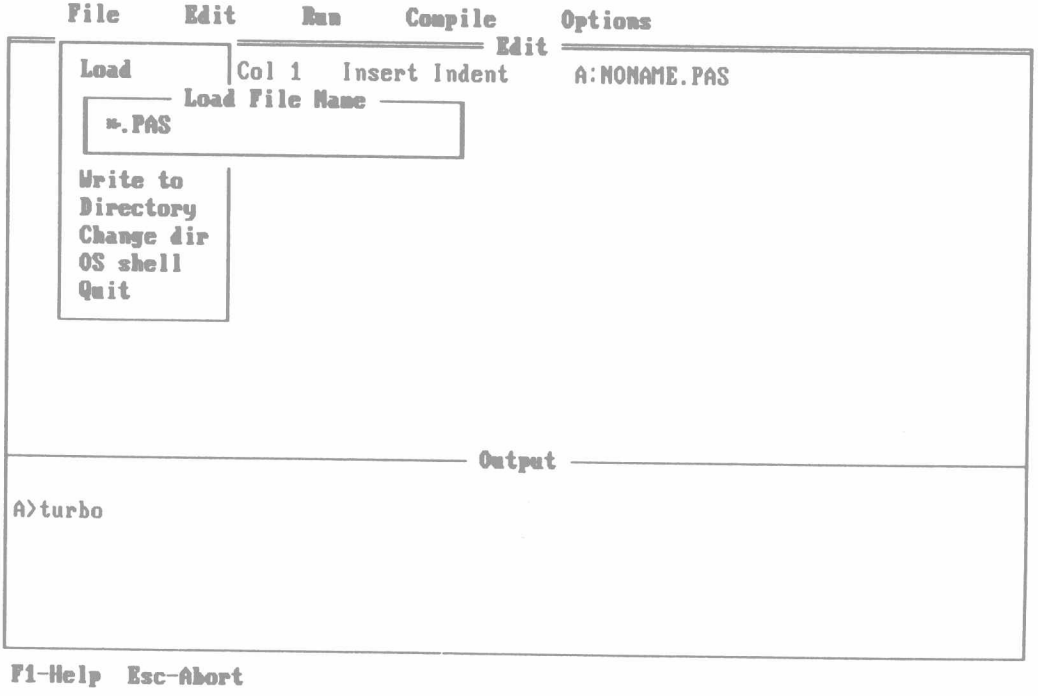

This is called the main screen. The main screen consists of the main menu bar, the edit window, the output window, and the function key summary line. While the main screen is displayed, only the main menu bar, edit window, or output window may be active. When you first enter the integrated environment, the edit window is active.

To use the integrated environment's many capabilities, you need to know how to activate different areas of the main screen. The F6 key selects which window is active, edit or output. Function F10 temporarily activates the main menu bar. To select a sub-menu from the main menu bar, press the left and right arrow keys to move the highlight between the words File, Edit, Run, Compile, and Options and then press enter. When you decide to leave the main menu (or any sub-menu), press the Escape key. To exit an environment, press the ALT key simultaneously with the X key. When you exit the environment, or change files to be edited, the integrated environment will prompt you to save any altered files.

The File sub-menu that you see below was displayed by pressing F10 to activate the menu bar, using the left and right arrow keys to move the highlight to the word File, and pressing the Enter key.




The Load sub-menu below was displayed by using the up and down arrow keys to highlight the word Load and pressing enter. As always, the sub-menu is displayed directly below the menu word used to invoke it, making it easier to identify where you are.



The following will describe the essential options available from the main menu bar. For more information on those options that are occasionally convenient, see Borland's Turbo Pascal Owner's Handbook.

## FILE OPTIONS

**Load:** This option prompts you for the name of a file. The named file is read into memory and becomes the current edit file. The file name may be one to eight characters in length. The file name suffix .PAS is automatically added to the file name by the Turbo system.

**Pick:** This option presents you with a list of file names that have been edited in the current session of the integrated environment.

**New:** Refreshes the editor back to editing the empty NONAME.PAS file.

**Save:** Save the current editor file to disk.

**Write to:** Save the current editor file to disk under a different name.

**Directory:** Prompts for Dir parameters and performs the directory command.

**Change dir:** Changes the current (default) directory. This can be used to choose a different default disk drive.

**OS shell:** Temporarily suspends the integrated environment to allow you to enter standard operating system commands. Just before you see the operating system prompt, a reminder will be displayed to type "Exit" to return to the integrated environment.

**Quit:** Terminates the integrated environment. If you have updated a file, you will be prompted to save the changes before the Turbo program terminates.

## COMPILE OPTIONS

**Compile:** Executes the compiler on the file currently being edited.

**Make:** Beyond the scope of this text.

**Build:** Beyond the scope of this text.

**Destination:** Switches compiler between putting object code into memory or onto a disk.

**Find error:** Locates position of a run-time error.

**Primary file:** Related to Make and Build options: beyond the scope of this book.

**Get info:** Displays a summary of the program and its last compile.

**Compiler:** Sets how the compiler will produce object code.

**Environment:** Sets various characteristics of the Turbo integrated environment, such as whether edited files are backed up and whether they are saved periodically.

**Directories:** Determines which directory special files will come from: beyond the scope of this text.

**Parameters:** Allows entering of compiler parameters: beyond the scope of this text.

**Load options:** Beyond the scope of this text.

**Save options:** Beyond the scope of this text.

29→ To enter a program with the Turbo Screen Editor, boot the system, start the Turbo system, and press F3 to load a file. The screen will be cleared, a status line will appear at the top of the Edit Window, and the current contents of the loaded file will be displayed. This message is called the editor status line, and it will remain on the screen as long as you are using the editor.

```
Line 1  Col  1     Insert       Indent     A:WORKFILE.PAS
```

The status line indicates the line number and the column number of the cursor (the little square that shows where you are typing). It also indicates that the editor is in the insert mode and the auto-indent mode. The last entry on the status line is the name of the file you are editing (the current file).

Anything you type will appear on the screen. To make corrections, you may use the backspace key (the left arrow key above the return key). To begin a new line, press the return key. You may use the cursor control keys to move the cursor around on the screen (see Figure 1–17 in the text on page 28). If you move the cursor into the middle of some text and begin typing, what you type will be inserted at that point.

On the IBM PC the Home key will move the cursor to the left end of a line, and the End key will move the cursor to the right end of a line. The Del key deletes the character that the cursor is resting on (in contrast, the backspace key deletes the character to the left of the cursor). The Ins key switches the editor between the insert mode and the overwrite mode. In overwrite mode anything you type will replace existing text (instead of being inserted into the existing text).

If the file contains more than one screen of text, you may move up and down through the text by pressing the PgUp and PgDn keys, respectively. These keys move to the next or previous screen-sized piece of the file.

The remaining editor commands involve the use of the Ctrl key. This key is used as a shift key on a typewriter is used. That is, while holding the Ctrl key down, you press another key and then release both of them. For example, the notation Ctrl-A means that while you hold the Ctrl key down, strike the A key once. Ctrl-Q and Ctrl-K are especially important in the Turbo editor: They are the extended-command keys. After pressing Ctrl-Q or Ctrl-K, you will press another letter to issue an extended command. For example, the

command to quit the editor is Ctrl-K-D. This notation means that you press the Ctrl-K combination, then release the Ctrl and K keys and press the D by itself. The following list summarizes the remaining Turbo editor commands.

| | |
|---|---|
| Move one word left | Ctrl-Left Arrow |
| Move one word right | Ctrl-Right Arrow |
| Move screen up one line | Ctrl-W |
| Move screen down one line | Ctrl-Z |
| Move to top of screen | Ctrl-Home |
| Move to bottom of screen | Ctrl-End |
| Move to top of file | Ctrl-PgUp |
| Move to bottom of file | Ctrl-PgDn |
| Delete word to right | Ctrl-T |
| Remove line | Ctrl-Y |
| Delete from cursor to end of line | Ctrl-Q-Y |
| Mark the start of a block | F7 |
| Mark the end of a block | F8 |
| Make current word as a block | Ctrl-K-T |
| Turn block on or off | Ctrl-K-H |
| Copy turned-on block to current cursor position | Ctrl-K-C |
| Move turned-on block to current cursor position | Ctrl-K-V |
| Delete turned-on block | Ctrl-K-Y |
| Move to beginning of block | Ctrl-Q-B |
| Move to end of block | Ctrl-Q-K |
| Write block to a disk file | Ctrl-K-W |
| Read disk file as a block | Ctrl-K-R |
| Tab | Ctrl-I |
| Undo changes to current line | Ctrl-Q-L |

There are several other commands this editor has to offer, but these few should be sufficient for your programming tasks. If you need more information, consult Borland's Turbo Pascal Owner's Handbook. Note also that the integrated environment has a substantial amount of reference information available via the F1 "Help" key. Press F1 twice, and you will activate a menu of help topics ranging from use of the editor and the integrated environment to characteristics of the Turbo Pascal compiler and the language it supports. Throughout this supplement Borland's Turbo Pascal Owner's Handbook is referred to. Often this information is also available via the F1 key.

After you decide to quit the editor, press the F2 key to save your work. If you do not save your program, all of your work will be lost when you turn the computer off.

30→ To compile a program in Turbo, press the F2 key to save the file, then press the F10 key to put the cursor on the main menu bar. Move the cursor over to the Run option and press enter. For a long program a message will appear on the screen that tells you how many lines have been compiled thus far. For a short program the compilation goes so quickly that the message barely has time to appear.

If there are no errors in the program, it will then start to run. If an error is found, a message will be displayed describing the error, and you will then be returned to the Turbo editor with the cursor positioned at the point where the error was detected. Note that the point where the error was detected is not necessarily the point where the error actually is.

However, you are guaranteed that the error is somewhere prior to that point. For example, an error that results from an incorrect definition may not be detected until you try to use the thing you've defined.

If the error is simple enough, you may fix it in the editor and try to run the program again. If the error is not simple to fix, save the file, and then take the time to figure out the problem before you start changing the program. Never fix a bug until you're sure you understand the problem that caused the error. Otherwise, you'll just make matters worse. After you have determined that you are done debugging the program, use the Compile option to perform the compile. Be sure that the Destination option of the Compile sub-menu says disk. Activate the Compile option of the Compile sub-menu and a new file will be created. The new file will contain your executable object code and will have the same name, but with the suffix .EXE.

31→ Always be sure that you've saved your program on disk and that you've removed the disk from the drive before you turn the power off. If you've changed a file and you forget to save it before quitting the Turbo system, a message appears that will remind you to save the file. Don't turn the power off before you quit the Turbo system—that way your edited file will never be lost.

## Chapter 2 Problem Solving, Syntax/Semantics, and Pascal Programs

46→ Turbo Pascal limits the length of identifiers to 126 characters, the first 63 of which must be unique. Because program lines are also limited to this length, you should never have an identifier that is too long. It is also unlikely that you will ever mistakely define two identifiers that are identical in their first 63 characters.

47→ Your identifiers may contain both upper- and lowercase letters. Upper- and lowercase letters are considered to be the same by the compiler (XyZ) is the same identifier as xYZ). Turbo Pascal also permits the underscore character (_) to be used in identifiers. The underscore is most often used to improve the readability of identifiers that are made up of more than one word. For example, you might use

```
Data_Item     Max_Temp          ID_Number
```

Keep in mind, however, that this is a nonstandard feature of Turbo Pascal and that it cannot be used with most other compilers. Thus if you write a program in Turbo Pascal that you will eventually move to another computer system, you should not use underscores in identifier names.

48→ Turbo Pascal provides three integer types: Integer and the nonstandard types LongInt and ShortInt. Their ranges are:

| TYPE | RANGE |
|---|---|
| Integer | –32768 to 32767 |
| LongInt | –2147483648 to 2147483647 |
| ShortInt | –128 to 127 |

(The reason for such seemingly strange maximum and minimum integer values relates to the fact that in the computer integers are represented as binary numbers.)

49→ Turbo Pascal also permits the use of the nonstandard types Byte and Word. These types have special properties that make them useful for advanced programming techniques on personal computers. Note that these types can take advantage of the hexadecimal constants available in Turbo Pascal.

| TYPE | DECIMAL RANGE | HEXADECIMAL RANGE |
|---|---|---|
| Byte | 0 to 255 | $00 to $FF |
| Word | 0 to 65535 | $0000 to $FFFF |

53→ Turbo Pascal provides two extensions to standard Pascal's CONST section: "typed constants," which might be better named "initialized variables," and the use of previously named constants within constant definitions. The following examples show each of these nonstandard capabilities. For more information, see Borland's Turbo Pascal Owner's Handbook.

```
CONST
  Pi = 3.14159;
  TwoPi = 2 * Pi;
  Radius: Integer = 4;

BEGIN
  Radius := Radius + 1
```

60→ Turbo Pascal allows comments to be surrounded by either the curly braces { and } or the combination of (* and *). These notations may not be mixed; a comment ends when its matching end notation is used.

```
(* This is *not* a valid comment
(* This    *is*   a valid comment} *)
```

Turbo Pascal allows more flexibility than does standard Pascal in the placement of the CONST and VAR sections. They may be placed in any order and they may be repeated. So a program may have any number of CONST and VAR sections, and the first section might be a VAR section.

66→ Turbo Pascal defines additional nonstandard reserved words and predeclared identifiers that are listed in the first two appendixes of this supplement.

81→ Refer to the appendices at the end of this supplement for the standard identifiers of Turbo Pascal. There are a large number of additions to the Pascal standard.

105→ In Turbo Pascal, Read works basically like the ISO standard. The exception is with variables of Turbo Pascal's nonstandard type String. When reading String variables, all characters up to the next <eoln> character are input. Those characters that will fit are placed into the String variable; those that will not are discarded. The file position is left at the <eoln> character; subsequent reads of String variables will input no characters and not change the file position.

The PC doesn't support true batch processing. It is intended to be used primarily in interactive mode. Instead, a batch program on the PC simply takes all of its input from a disk file and writes all of its output to either a disk file or the printer. This feature will be covered in the next section.

108→ Turbo Pascal requires that the nonstandard procedure Assign be called to match a system file name with a corresponding name in the program. Assign has two parameters. The first parameter is the name that will be used to refer to the file within the program. The second parameter is a string that is the system's name for the file. For example, if we have a file on disk called WEEKSPAY.DAT that the program will refer to as PayFile, then we use the following Assign statement to link the two names together:

```
Assign(PayFile, 'WEEKSPAY.DAT');
```

Notice that the second name is a string constant—it is enclosed in single quotation marks. Quotation marks are used because the format for PC-DOS file names is different from the format for Pascal identifiers. In Chapter 12 we will introduce string variables—the second parameter of the Assign statement may also be a string variable.

The Assign statement must come before any other statement that tries to access the file specified in the first parameter. Thus the Assign statement must also precede the Reset or Rewrite statement for a file. Assign may not be applied to the files Input and Output.

At the end of a program Turbo Pascal requires that the nonstandard procedure Close be called for files other than Input or Output. For example, you might have

```
Close(PayFile);
```

If this procedure is not used, a portion of the data written at the end of the file may be lost. Close may not be applied to the files Input and Output.

For Turbo Pascal, then, there are six steps involved in using files other than Input or Output:

1. List the names in the program heading.
2. Declare the names to be of type Text.
3. Assign the names to system files.
4. Reset or rewrite the files as appropriate.
5. Use the file name as the first parameter of each Read, Readln, Write, or Writeln.
6. Close the files at the end of the program.

A file may be closed before the end of the program if, for example, it has been entirely processed and is no longer needed. However, an error will result if you try to assign a new system file name to a file that is already in use.

Turbo Pascal also defines several additional nonstandard predefined files. The only one that is really of interest here is Lst, which is the printer. Lst is treated just like Output, except that you must include it as the first parameter of any Write or Writeln statement that you want to print on the printer. For example,

```
Writeln(Lst, 'This gets printed.');
```

will cause the message 'This gets printed.' to be typed on the printer rather than on the screen.

Because Lst is predefined in the printer unit, you don't have to declare it to be of type Text, you don't have to assign a system file name to Lst, and you don't have to rewrite or close it. All you have to do to use Lst is to place the following uses clause after the program heading but before any other declarations:

```
uses Printer;
```

## Chapter 4 Selection

139→ Turbo Pascal defines an additional nonstandard Boolean operator called XOR, which stands for eXclusive OR. The result of this operation is true when either, *but not both*, of its operands are true. If X and Y are Boolean variables, then

```
X XOR Y
```

is equivalent to

```
X <> Y
```

The latter form has the advantage that it conforms to the ISO standard for Pascal and may thus be used on any system.

Turbo Pascal specifies that it will short-circuit evaluations of AND in boolean expressions. That is, if the left expression of an AND expression is false, the right expression will not be evaluated. So with Turbo Pascal,

```
(X <> 0) AND ((Y / X) <> 1)
```

will not cause a run-time error: if X is zero, (X<>0) is false, so the expression that uses X as a divisor is not evaluated. Since standard Pascal does not specify whether AND's should be short-circuited, some compilers would allow the above expression to evaluate Y/0, causing a run-time error. If you want portable code, don't write expressions depending on short-circuiting of AND's.

Note also that NOT, AND, OR, and XOR are also logical operators on integers. These operators may allow otherwise unacceptable expressions to be compiled successfully. If you want further information on these operators, consult Borland's Turbo Pascal Owner's Handbook.

## Chapter 5 Looping

177→ If the program asks for input during execution of an infinite loop, you can then press Ctrl-C to stop it. Otherwise, you will have to press the Ctrl-Alt-Del key combination to boot the operating system. In this event, any changes you've made will be lost, and you will have to restart the Turbo system. Thus it's always a good idea to save your work file before running a program that you've just changed.

Occasionally, a program will get so wrapped up in itself that even Ctrl-Alt-Del will fail to stop it. You must then turn the PC off, wait about 5 seconds, and turn the machine back on.

## Chapter 6 Procedures

Turbo Pascal 4.0 implements procedures according to an extension of the ISO standard. The differences have to do with some nonstandard data types and hooks into the operating system that are used for advanced programming on the PC. These features are beyond the scope of this book and will not be discussed here.

## Chapter 7 Value Parameters and Nested Scope

273→ Turbo Pascal's uses clause modifies the book's scope rules to be:

1. The scope of an identifier includes all of the statements following its definition, within the block containing the definition. This includes nested blocks, except as noted in rule 2. The definition of an identifier may occur directly in the code or in an included file or uses clause.
2. The scope of an identifier does not extend to any nested block that contains a locally defined identifier with the same spelling. If two uses clauses define identifiers with the same spelling, the earlier identifier can be referenced only by qualifying it with its unit name.
3. The scope of a formal parameter is identical to the scope of a local variable in the same procedure.

An identifier is qualified by its unit name by placing the unit name and a period before the identifier. For example, if both Unit1 and Unit2 define an identifier 'A':

```
USES Unit1, Unit2;
BEGIN
   Unit1.A
.
.
```

then Unit1.A refers to the variable A declared in unit Unit1.

## Chapter 8 Functions, Precision, and Recursion

320→ The range of allowable exponents for Real numbers in Turbo Pascal is –38 through +38. Real numbers have 11 digits of precision in Turbo Pascal. There are four variations on the type Real in Turbo Pascal that require special hardware to be installed on your PC. The special hardware makes Real calculations execute much faster than standard Turbo Pascal code run on a standard PC. For details, see Borland's Turbo Pascal Owner's Handbook.

## Chapter 9 Sets and Additional Control Structures

334→ In Turbo Pascal a set may contain up to 256 elements. The base type of a set must have ordinal values in the range 0 to 255. Thus, for example, the declaration

```
VAR NotAValidSet: SET OF 100..300;
```

is illegal because the ordinal value of 300 is greater than 255.

350→ Just before the END of the CASE statement, Turbo Pascal allows you to insert ELSE as a case label list. The statement(s) associated with the ELSE will be executed if the case selector does not equal any of the values in the case label lists. For example, you might have

```
CASE Grade OF
  'A', 'B' : Write('Good Work');
  'C'      : Write('Average Work');
  'D', 'F' :
    BEGIN
      Write('Poor Work');
      NumberInTrouble := NumberInTrouble + 1
    END;
       ELSE       Write('Grade is not a legal letter grade')
END; (* Case *)
```

Notice that there is no colon (:) after the ELSE. In this sense it is like the ELSE branch of an IF-THEN-ELSE. Although the ELSE feature may be convenient, it is nonstandard and will not work under most other Pascal systems. Another nonstandard Turbo Pascal feature is that a semicolon may appear preceding the END of a CASE statement.

## Chapter 10 Simple Data Types

370→ Turbo Pascal allows a character to be represented by its ordinal value preceded by a pound sign. For example, #48 is the same as '0'. These characters can form strings by simply placing them next to other strings with no blanks or other characters in between. So #48'A'#49'B'#50 is the same as '0A1B2'.