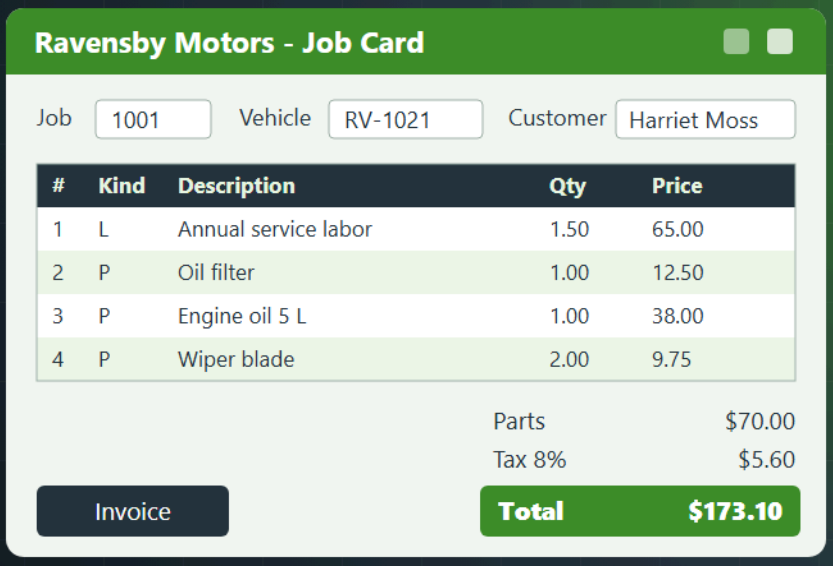
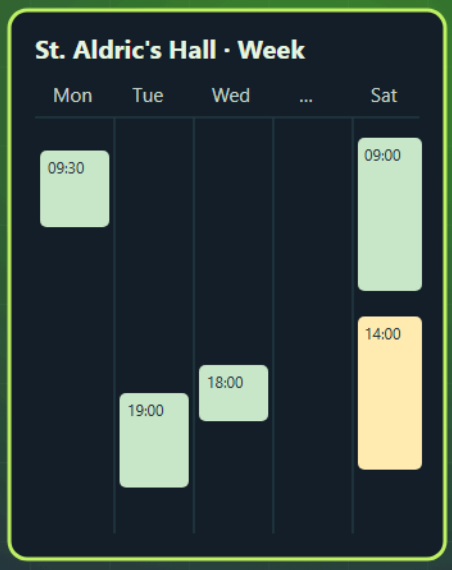





```
SELECT books
SET ORDER TO title
IF SEEK("THE")
  SCAN WHILE UPPER(title) = "THE"
    ? book_id, title
  ENDSCAN
```

1 · Library Loans
2 · Club Membership & Fees
3 · Hall Bookings
4 · Garage Job Cards
5 · Shop Inventory
6 · Studio Timesheets

2000000010038 · RM-00042 · READ EVENTS

LEARN BY BUILDING

Visual FoxPro by Project

Learn VFP 9 by building six complete business applications

Tables · Indexes · SQL · Forms · Grids · Reports · Classes
Buffering · Transactions · CSV & Excel · Multi-user · EXE

QUICK START · SIX PROJECTS · ONE TOWN

THENEXBOOKS

Visual FoxPro by Project

Learn VFP 9 by Building Six Complete Business Applications
From an empty folder to a working EXE — six times, one town, step by step

First Edition

THENEXBOOKS

Visual FoxPro by Project
Learn VFP 9 by Building Six Complete Business Applications

First Edition, 2026

Copyright © 2026 thenexbooks (thenexbooks.gumroad.com). All rights reserved.

No part of this publication may be reproduced, distributed, or transmitted in any form or by any means without the prior written permission of the publisher, except for brief quotations in reviews and certain other non-commercial uses permitted by copyright law. The code examples may be used freely in your own projects.

The information in this book is provided "as is" without warranty of any kind. While every precaution has been taken in its preparation, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein. Visual FoxPro 9.0 Service Pack 2 was the final release of the product, and Microsoft's extended support for it ended in January 2015; you need your own licensed copy to follow the examples. Back up your data before running examples that change or delete it.

Microsoft, Visual FoxPro, Visual Studio, Excel, Windows, and SQL Server are trademarks of the Microsoft group of companies. Other product names are trademarks of their respective owners. Their use in this book is for identification only and does not imply endorsement. This book is independent and is not affiliated with, sponsored by, or endorsed by Microsoft Corporation.

Ravensby and all of its organizations — Ravensby Library, Ravensby Rovers, St. Aldric's Hall, Ravensby Motors, Ravensby Corner Shop, and Ravensby Design Studio — are fictional, as are their people, books, members, vehicles, products, bookings, and all sample data. Phone numbers use the fictional 555-01xx range and email addresses use reserved example domains.

Preface

Some tools are best learned by reading about them. Visual FoxPro is not one of them. It is a programming language, a fast database engine, and a set of visual designers for forms, reports, menus, and classes, all in one environment — and it rewards the person who sits down, types a command, and watches what happens. The fastest way to become comfortable with it is to build something real, then build something a little bigger, and keep going.

That is exactly what this book does. After a short quick start, every Part builds complete small business applications from an empty folder to a working EXE. Each project solves a believable problem for a believable organization, and each one introduces a new set of skills at the moment you need them: indexes when a librarian needs to find a book quickly, rules and triggers when a sports club wants clean member data, date and time logic when a hall must never be double-booked, buffering and transactions when a garage invoice must be all-or-nothing, file import and Excel export when a shop owner lives in spreadsheets, and locking, menus, error logging, and deployment when a design studio needs a shared, multi-user timesheet system.

By the end you will have built six applications, and — more importantly — you will know how to plan, build, test, and ship the next one on your own.

Who This Book Is For

- **Beginners to Visual FoxPro**, including people who have never written a database application before. No prior FoxPro knowledge is assumed; every command and term is explained the first time it appears.
- **Office developers and power users** coming from Excel or Access who want a fast, programmable desktop database tool.
- **Junior developers at organizations that still run VFP applications** who want to learn by building, not just by reading old code.
- **Experienced FoxPro users** who want a structured, hands-on refresher of VFP 9 techniques through complete worked projects.

What You Will Learn

- The VFP 9 environment: the Command Window, the language, tables, Xbase commands, SQL, programs, and the Project Manager.
- Free tables, indexes, SEEK, and fast lookups — and when to move up to a database container with primary keys, relations, rules, triggers, and stored procedures.
- Forms of every kind: simple search screens, data-bound forms, grids and combo boxes, master-detail screens with private data sessions, and your own container classes.
- Reports: quick reports, grouping and totals, and business documents such as an invoice.

- Real business logic: loan rules and fines, age-based fees and family discounts, booking clashes and weekly series, invoice totals with tax, barcode check digits, and timesheet limits.
- Safe data changes with buffering and transactions, and multi-user work with locking, optimistic buffering, and update conflicts.
- Getting data in and out with CSV files, text merge, low-level file functions, and Excel automation.
- Building complete applications: main programs, menus, READ EVENTS, base classes, error handling and logging, building EXEs, deploying runtime files, and updating installed copies.

How This Book Is Organized

Part I — Quick Start (Chapters 1–5) sets up VFP 9 and teaches the essentials with a small town-directory example: the Command Window, the language, tables and Xbase commands, SQL, programs, the Project Manager, and your first EXE. **Part II — Project 1: Ravensby Library Loans** (Chapters 6–9) builds a loans tracker with free tables, indexes, SEEK, loan rules, first forms, and a simple report. **Part III — Project 2: Ravensby Rovers Membership and Fees** (Chapters 10–14) introduces the database container, rules, triggers, validation, fee calculations, grids, combo boxes, and grouped reports. **Part IV — Project 3: St. Aldric's Hall Bookings** (Chapters 15–18) works with dates and times, booking clashes, weekly series, a calendar built from your own container classes, and text-merge confirmations. **Part V — Projects 4 and 5** (Chapters 19–23) builds a garage job-card and invoicing app with a business object, buffering, transactions, a master-detail form, and an invoice report, then a corner-shop inventory with barcode lookups, CSV import and export, and Excel automation. **Part VI — Project 6: Ravensby Design Studio Timesheets** (Chapters 24–28) builds a multi-user application with shared data, an application object, menus, base classes, locking and conflict handling, error logging, and deployment. The appendices provide a cheat sheet, project templates and reference tables, a release checklist, and a glossary.

Every chapter ends with **Practice** exercises and a **Key Takeaways** box for quick review. Later projects build on skills from earlier ones, but each project starts in its own empty folder, so you can also jump to a project that matches what you need.

The Running Example: Ravensby

Ravensby is a fictional market town, and every project in this book serves one of its organizations. Margaret Oduya at **Ravensby Library** needs to know which books are out and who owes fines. Dan Whitlow, secretary of the **Ravensby Rovers** sports club, wants member records he can trust and a clear list of unpaid fees. Clara Benn runs the bookings at **St. Aldric's Hall** and must never let two groups book the same room at the same time. Frank Maddox at **Ravensby Motors** needs job cards and invoices that add up. Asha Patel at the **Ravensby Corner Shop** wants to scan barcodes, track stock, and swap price lists with her supplier. Joel Ambrose at the **Ravensby Design Studio** needs a

timesheet system his whole team can use at the same time. Each project opens with that person's brief, and the sample data is small enough to check by hand: every number shown in this book was produced by running the code in Visual FoxPro 9.

Versions and Conventions

Examples target **Visual FoxPro 9.0 with Service Pack 2** (SP2), the final release of the product, running on Windows. Microsoft no longer sells or supports Visual FoxPro, so you need an existing licensed copy; never download it from unofficial sites. Community add-ons are mentioned only as options. The examples assume US settings — `SET DATE AMERICAN` with `SET CENTURY ON`, so dates appear as `03/16/2026`, and currency as `$45.00`. The book's "today" in worked examples is Monday, March 16, 2026.

> **TIP**
> A recommended practice or a technique that makes your work easier.

> **NOTE**
> Background information, version differences, or a useful aside.

> **WARNING**
> A common mistake or something that can cause data loss, errors, or confusing results.

Commands, code, and output appear in `monospace`; VFP keywords are written in uppercase and variables use a short type prefix such as `lcName` or `lnTotal`. Each project lives in its own folder under `C:\Ravensby\` with the same layout — `data\`, `progs\`, `forms\`, `reports\`, and so on — explained in Chapter 1 and summarized in Appendix B. Menu commands are written in bold with arrows, such as **Tools > Options**.

Contents

Preface

PART I — QUICK START

1 Welcome to Visual FoxPro: Six Apps, One Town, and Setting Up VFP 9

2 The Command Window and the Language: Variables, Expressions, and Functions

3 Tables and Xbase: Creating, Browsing, and Changing DBF Data

4 SQL in Visual FoxPro: SELECT, Cursors, Joins, and Changing Data

5 Programs, the Project Manager, and Your First EXE

PART II — PROJECT 1: RAVENSBY LIBRARY LOANS

6 Planning the Library App: Folders, Free Tables, and Sample Data

7 Indexes and SEEK: Fast Lookups for Books and Members

8 Loan Rules in Code: Checking Out, Returning, and Fines

9 Your First Forms: Book Search, Check-Out, the Overdue Report, and library.exe

PART III — PROJECT 2: RAVENSBY ROVERS MEMBERSHIP AND FEES

10 A Database Container for the Club: Primary Keys, Relations, and Stored Procedures

11 Rules, Triggers, and Validation: Keeping Member Data Clean

12 Calculating Fees: Age Categories, Family Discounts, and Payments

13 Grids and Combo Boxes: The Members and Payments Screens

14 Reports for the Club: Roster, Fees Owing, and Grouped Totals

PART IV — PROJECT 3: ST. ALDRIC'S HALL BOOKINGS

15 Dates and Times in VFP: Modeling Hall Bookings

16 Booking Rules: Opening Hours, Clashes, Prices, and Weekly Series

17 A Week Calendar with Containers and Your Own Classes

18 Confirmations, Week Reports, and Finishing the Hall App

PART V — PROJECTS 4 AND 5: RAVENSBY MOTORS AND THE CORNER SHOP

19 Ravensby Motors: Job Cards, a Business Object, and Invoice Totals

20 Buffering and Transactions: Invoicing a Job Safely

21 The Job Card Form: Master-Detail Screens, Private Data Sessions, and the Invoice Report

22 Ravensby Corner Shop: Barcodes, Check Digits, and a Scan-to-Sell Screen

23 Data In and Out: CSV Import and Export, and Excel Automation

PART VI — PROJECT 6: RAVENSBY DESIGN STUDIO TIMESHEETS

24 Planning a Multi-User App: Shared Data, Rules, and the Application Object

25 The Application Shell: main.prg, Menus, READ EVENTS, and Base Classes

26 Many Users, One Database: Locking, Optimistic Buffering, and Update Conflicts

27 When Things Go Wrong: Error Handling, Logging, and Debugging

28 Building, Deploying, and Updating the Timesheet App — and Where to Go Next

APPENDICES

A VFP 9 Cheat Sheet

B Project Templates and Reference

C Release and Deployment Checklist

D Glossary

PART I

Quick Start

Setting up Visual FoxPro 9 and learning the essentials by doing: the Command Window, the language, tables and Xbase commands, SQL and cursors, programs, the Project Manager, and building your first EXE — all with a small Ravensby town directory.

1 · Welcome to Visual FoxPro

2 · The Command Window and the Language

3 · Tables and Xbase

4 · SQL in Visual FoxPro

5 · Programs, Projects, and Your First EXE

CHAPTER 1 · PART I — QUICK START

Welcome to Visual FoxPro: Six Apps, One Town, and Setting Up VFP 9

In this chapter you meet Visual FoxPro, set up a tidy workspace for the whole book, take a tour of the development environment, and run your very first program. You also meet the small town of Ravensby, whose six organizations will give us six complete applications to build together.

What Visual FoxPro Is

Visual FoxPro (VFP for short) is three things packed into one product. First, it is a **programming language**: you write commands and functions, store values in variables, make decisions with `IF`, and repeat work with loops, just as you would in any other language. Second, it is a **database engine**: the language can create tables, add and change records, search them with indexes, and query them with SQL, all without a separate database server. Third, it is a set of **visual designers**: you draw forms, reports, menus, and classes with the mouse, set their properties in a window, and attach code to the events they raise.

That combination is the reason VFP became popular for business software. In many tools the language, the data, and the screens live in different worlds, and much of your time goes into gluing them together. In VFP a table is a first-class citizen of the language. You can type `USE books` and the table is open; you can type `REPLACE status WITH "L"` and the current record changes; you can write a SQL `SELECT` in the middle of a program and get back a cursor you can loop over, bind to a grid, or print in a report. The language understands records, fields, dates, and money natively.

The tables VFP uses are files with the extension `.dbf` (for "database file"). The format goes back to the dBASE and FoxPro products of the 1980s and early 1990s, which is why you will hear VFP described as an **Xbase** language. "Xbase" is the family name for languages that share those roots: commands such as `USE`, `LIST`, `SKIP`, and `REPLACE` that work directly on the records of an open table. VFP added an object-oriented layer (classes, inheritance, forms as objects), a full SQL dialect, and a database container that groups tables with rules and relations. You will use all of these in this book.

A short, honest history

We want you to start with accurate expectations, so here are the facts that matter. Visual FoxPro 9.0 was released by Microsoft in December 2004. Service Pack 2 for VFP 9 followed in 2007 and was the final release of the product. In March 2007 Microsoft announced that there would be no Visual FoxPro 10. Extended support for VFP 9 ended in January 2015. In this book we write the version as **VFP 9 (SP2)**, because that is the release we target.

In practical terms this means:

- **Microsoft no longer sells VFP.** You need an existing licensed copy, for example from an old MSDN subscription or from your company's software media. If your workplace runs VFP applications, ask your IT department whether a licensed copy is available for you.
- **Never download VFP from unofficial websites.** Installers passed around on the internet may be tampered with, and using them is not a legal way to obtain the product. This book never asks you to download anything.
- **No new features will arrive from Microsoft.** What you learn here is stable: the language, designers, and file formats will not change under you.
- **The community is still active.** Developers continue to share tools and libraries as community projects on VFPX. We mention a few of them later in the book, always as options rather than requirements, and we make no promises about their versions or features.

Why learn a product whose last release is from 2007? Because a great deal of working business software was written in VFP and is still in daily use. Companies need people who can read it, maintain it, extend it, and sometimes move its data to newer systems. VFP is also a pleasant way to learn how database applications work: the distance between an idea ("show me all overdue books") and a working result is short, and habits such as keeping business rules in one place and testing with repeatable sample data carry over to any language you use later.

> **NOTE**
>
> The verification PC used for this book runs VFP 9 with Service Pack 1. Every language feature we use behaves the same in SP1 and SP2, so you can follow along with either. Where your screen might differ slightly (a version number, a dialog title), we say so.

Who This Book Is For, and How It Works

This book is for people who want to *build things* with Visual FoxPro. Maybe you are an office developer who has outgrown Excel or Access, a student who has been handed a VFP assignment, a junior developer at a company that still runs VFP systems, or a hobbyist who is curious about a classic tool. We do not assume you have programmed a database before. Every term is explained the first time it appears, and every command shows you what it prints.

The book is **project-based**. Part I (Chapters 1 to 5) is a quick start: the development environment, the language, tables, SQL, and building your first EXE. From Part II onward, each Part builds a complete small application, step by step, from an empty folder to a working program you could hand to a real user. Each project introduces new skills at the moment the project needs them. You will not read a long abstract tour of every command; instead, you will meet indexes when the library needs to find a book quickly, transactions when the garage needs to invoice a job safely, and locking when the design studio has several people saving timesheets at once.

> **TIP**
>
> Type the examples yourself rather than copying them. Typing slows you down just enough to notice details such as a missing comma or a misspelled field name, and the mistakes you make (and fix) teach you more than the code that works the first time.

Meet Ravensby: Six Organizations, Six Applications

Our running example is the fictional market town of **Ravensby**. It has a library, a football club, a church hall that is rented out for events, a garage, a corner shop, and a small design studio. Each of them has a problem that a modest database application can solve, and each of them has a person who will brief you on what they need. You play the role of the developer who builds all six applications.

Project	Organization	Who briefs you	Chapters	Main new skills
Warm-up	Town directory	(you)	1–5	IDE, language, tables, SQL, first EXE
1. Library Loans	Ravensby Library	Margaret Oduya	6–9	Free tables, indexes, SEEK, rules in code, first forms and report
2. Membership and Fees	Ravensby Rovers	Dan Whitlow	10–14	Database container, rules, triggers, grids, grouped reports
3. Hall Bookings	St. Aldric's Hall	Clara Benn	15–18	Dates and times, clash checks, your own classes, text merge
4. Job Cards and Invoices	Ravensby Motors	Frank Maddox	19–21	Business objects, buffering, transactions, private data sessions
5. Corner Shop	Ravensby Corner Shop	Asha Patel	22–23	Barcodes, CSV import and export, Excel automation
6. Timesheets	Ravensby Design Studio	Joel Ambrose	24–28	Multi-user data, menus, error handling, deployment

The projects grow in ambition. The library app uses simple free tables and a handful of forms. The club app introduces a database container with rules that refuse bad data. The hall app deals with dates and times and draws a week calendar with its own classes. The garage app keeps invoices consistent with buffering and transactions. The shop app talks to barcode scanners, CSV files, and Excel. The studio app is used by several people at the same time on a shared network folder, so it needs locking, conflict handling, a proper menu-driven shell, error logging, and an installer.

Along the way you will also meet **Sam Kerr**, the library assistant who uses the check-out screen every day, and you will build a small town directory in Part I that lists all six organizations and their contacts. That directory is our practice data for Chapters 2 to 5, so you will become very familiar with names such as "Ravensby Motors" and "St. Aldric's Hall" before the real projects begin.

Installing and Starting VFP 9

Because installation media differ (an MSDN disc image, a company network share, a boxed retail copy), we describe the process in general terms. Follow the instructions that came with your licensed copy.

1. **Run the setup program** from your VFP 9 media. Setup usually offers to install prerequisites first and then the product itself. Accept the default options unless your IT department tells you otherwise.
2. **Apply the service pack.** If your media contains VFP 9 without a service pack, install SP2 from the same licensed source. Many companies keep the service pack alongside the original media.
3. **Find the program.** On 64-bit Windows, VFP 9 (a 32-bit program) typically installs into a folder such as `C:\Program Files (x86)\Microsoft Visual FoxPro 9\`, and the program file is `vfp9.exe`. Setup normally adds a shortcut to the Start menu.
4. **Start VFP.** The first start may take a moment while VFP creates its resource file (`FOXUSER.DBF`), a small table where it remembers window positions and other preferences for you.

> **WARNING**
>
> Do not install VFP or keep your projects inside `C:\Program Files` or `C:\Program Files (x86)` for your own work. Recent versions of Windows protect those folders, and programs that try to write files there may have their changes silently redirected elsewhere or refused. Keep all your work in a folder you own, such as the `C:\Ravensby\` folder we create in a moment.

A Tour of the Development Environment

When VFP starts, you see one large window: the **main VFP window**. Inside your programs you can refer to it as `_SCREEN`, and you will learn to change its caption in Chapter 25. For now, here is a rough sketch of what you see:

```
+-------------------------------------------------------------------+
| Microsoft Visual FoxPro                                  _  [] X  |
+-------------------------------------------------------------------+
| File  Edit  View  Format  Tools  Program  Window  Help            |
+-------------------------------------------------------------------+
| [New][Open][Save][Print] ... [Run !] ... [Command Window][Data]   |
+-------------------------------------------------------------------+
|                                                                   |
|    +-- Command ------------------------------+                   |
|    | ? "Hello"                               |                   |
|    | ? DATE()                                |                   |
|    | _                                       |                   |
|    +-----------------------------------------+                   |
|                                                                   |
|  (output from ? appears here, on the main window background)      |
|                                                                   |
```

```
+-------------------------------------------------------------------+
| status bar: messages, record numbers, Ins/Num indicators          |
+-------------------------------------------------------------------+
```

The parts you will use most often are described below. Do not worry about memorizing them; you will meet each one again when a project needs it.

The menu bar and the Standard toolbar

The menu bar changes slightly depending on what you are doing. When you edit a program, a **Program** menu offers commands to run it; when you design a form, a **Form** menu appears. The **Standard** toolbar under the menu gives one-click access to New, Open, Save, Print, and a **Run** button (an exclamation mark) that runs the program or form you are editing.

The Command Window

The **Command Window** is the heart of VFP development. You type a command, press **Enter**, and VFP runs it immediately. It is a calculator, a database console, a test bench, and a launcher for every designer, all in one small window. If you close it by accident, choose **Window > Command Window** or press **Ctrl+F2** to bring it back.

The Command Window remembers everything you type. Use the arrow keys or the mouse to move up to an earlier line, change it if you like, and press Enter to run it again. You can also select several lines with the mouse and press Enter to run them all in order. Chapter 2 is devoted to using the Command Window as a playground for the language.

The Properties window (a preview)

When you design a form or a class, the **Properties window** lists every property of the selected object (its caption, size, color, the field it shows) and every event and method you can write code for. You open it from **Window > Properties** or by right-clicking an object in a designer. We use it from Chapter 9 onward.

The Data Session window (a preview)

The **Data Session** window (**Window > Data Session**, or type `SET` in the Command Window) shows which tables are open, in which work areas, and with which aliases. It is a quick way to check "what is open right now?" while you experiment. Chapter 3 explains work areas and aliases, and Chapter 21 explains why each form can have a private set of open tables, called a data session.

The Project Manager (a preview)

A real application is made of many files: programs, forms, reports, menus, class libraries, and tables. The **Project Manager** keeps track of them in a project file (`.pjx`) and builds them into a single executable (`.exe`). It has tabs that group files by kind: All, Data, Documents, Classes, Code, and Other. You create your first project and EXE in Chapter 5.

Your First Commands

Let us make VFP do something. Click in the Command Window and type the following lines, pressing Enter after each one. The question mark command (`?`) evaluates whatever follows it and prints the result on a new line of the main window.

Command Window

```
? "Hello from Ravensby!"
? 2 + 2
? VERSION()
? DATE()
```

You should see something like this (your date will be today's date, and the build number at the end of the version string depends on your service pack):

Output

```
Hello from Ravensby!
 4
Visual FoxPro 09.00.0000.5815 for Windows
10/08/2026
```

Four lines, four lessons. Text in quotes is a **character string** and is printed exactly as written. The expression `2 + 2` is calculated before printing, and numbers are printed right-aligned, which is why there is a space before the `4`. `VERSION()` is a **function**: a named piece of built-in logic that returns a value, written with parentheses. `DATE()` returns today's date, displayed in the date format that is currently set.

To clear the main window, type `CLEAR` and press Enter. Nothing is lost; it only wipes the output area. To find help on any command, type `HELP` followed by the command name (for example `HELP DATE`) or press **F1** while the cursor is on a word in the Command Window. The help file that ships with VFP 9 is thorough and full of examples, and you will use it often.

> **NOTE**
>
> Throughout this book dates are shown in US format, `MM/DD/YYYY`, with a four-digit year. If your copy of VFP shows dates differently, for example `08.10.26`, do not worry: Chapter 2 explains `SET DATE` and `SET CENTURY`, and every project sets the format it needs at start-up. The worked examples treat **Monday 03/16/2026** as "today" so that your results match ours exactly.

Setting Up the C:\Ravensby Workspace

Good habits start with a tidy folder structure. Every project in this book lives in its own folder under `C:\Ravensby\`, and every project folder has the same subfolders. When you know where everything goes, you never have to wonder where a form or a table ended up.

```
C:\Ravensby\
   QuickStart\        Part I warm-up: the town directory
   Library\           Project 1 - Ravensby Library Loans
   Rovers\            Project 2 - Ravensby Rovers Membership and Fees
   Hall\              Project 3 - St. Aldric's Hall Bookings
   Garage\            Project 4 - Ravensby Motors Job Cards and Invoices
   Shop\              Project 5 - Ravensby Corner Shop
   Timesheet\         Project 6 - Ravensby Design Studio Timesheets

C:\Ravensby\<Project>\
   data\              tables (.dbf), indexes (.cdx), database containers (.dbc)
   progs\             programs (.prg)
   forms\             forms (.scx)
   libs\              class libraries (.vcx)
   reports\           reports (.frx)
   menus\             menus (.mnx)
   (import\ export\ output\ when a project needs them)
   project file (.pjx) and the finished .exe in the project root
```

You can create the folders in Windows Explorer, but VFP can do it too, which is a nice first taste of commands that change the file system. The `MD` command (short for "make directory") creates a folder, and `CD` ("change directory") makes a folder the current one. Type these lines in the Command Window:

Command Window

```
MD C:\Ravensby
MD C:\Ravensby\QuickStart
MD C:\Ravensby\QuickStart\data
MD C:\Ravensby\QuickStart\progs
CD C:\Ravensby\QuickStart
? SYS(5) + CURDIR()
```

Output

```
C:\RAVENSBY\QUICKSTART\
```

`SYS(5)` returns the current drive and `CURDIR()` the current folder, so together they show where you are. VFP prints folder names in uppercase; Windows file names are not case-sensitive, so `C:\Ravensby` and `C:\RAVENSBY` are the same folder. If a folder already exists, `MD` reports an error; that is harmless, and you can simply move on. We create the project folders for the other projects when we reach them.

The **current folder** matters because every path in this book is written **relative to the project root**. When we write `USE data\orgs`, VFP looks for `data\orgs.dbf` inside the current folder. That is why each working session starts with the same line:

Command Window

```
CD C:\Ravensby\QuickStart
```

Relative paths are more than a convenience. When you later copy a finished project to another PC, or to a different drive, nothing inside the code has to change, because no program mentions `C:\Ravensby` directly.

Making the folder the default in Tools > Options

VFP can start in your workspace automatically. Choose **Tools > Options**, open the **File Locations** tab, select **Default Directory**, click **Modify**, tick "Use Default directory", and choose `C:\Ravensby`. Then click **Set as Default** before **OK**. Without "Set as Default", most changes in the Options dialog last only for the current VFP session; with it, VFP saves them and uses them every time it starts.

```
+-- Options ------------------------------------------------------+
| View | General | Data | Remote Data | File Locations | Forms ... |
+-------------------------------------------------------------------+
|  File Type                 Location                               |
|  Default Directory         C:\RAVENSBY                            |
|  Help File                 ...                                    |
|  Resource File             ...                                    |
|                                                                   |
|                                  [Modify...]                      |
+-------------------------------------------------------------------+
|             [Set as Default]   [OK]   [Cancel]   [Help]           |
+-------------------------------------------------------------------+
```

While the dialog is open, glance at the **Data** tab: its "Open exclusive" setting is on in a fresh installation, which surprises most beginners. Chapter 3 explains why.

> **TIP**
> Many experienced developers prefer to leave the Options dialog alone and put every setting their code needs into the code itself, with commands such as `SET DATE` and `SET EXCLUSIVE`. That way a program behaves the same on every PC, no matter how the IDE there is configured. We follow that rule in every project: each one has start-up code that sets its own environment.

Your First Program: hello.prg

Typing commands one at a time is perfect for experiments, but real work lives in **programs**: text files with the extension `.prg` that contain a list of commands. VFP runs them from top to bottom. Let us write the smallest possible program. Make sure you are in the QuickStart folder, then type:

Command Window

```
CD C:\Ravensby\QuickStart
MODIFY COMMAND hello
```

`MODIFY COMMAND` opens the program editor on `hello.prg`, creating the file if it does not exist yet. You do not need to type the extension; VFP adds `.prg` for you. Type the program below into the editor window:

C:\Ravensby\QuickStart\hello.prg

```
* hello.prg - our first Visual FoxPro program
? "Hello from Ravensby!"
```

The first line is a **comment**. Any line that starts with an asterisk (`*`) is ignored by VFP; it is there for human readers. The second line is the same `?` command you typed earlier. Save the file and close the editor with **Ctrl+W** (or choose **File > Save** and then close the window). Now run it:

Command Window

```
DO hello
```

Output

```
Hello from Ravensby!
```

Congratulations: you have written and run a Visual FoxPro program. If you look in the folder now, you will find two files, `hello.prg` and `hello.fxp`. The `.fxp` file is the **compiled** version of your program. When you run `DO hello`, VFP checks whether the source is newer than the compiled file, compiles it if needed, and runs the compiled code. You never need to edit an `.fxp` file, and it is safe to delete one; VFP simply recreates it on the next run.

You can also run a program that is open in the editor by clicking the **Run** button (the exclamation mark) on the Standard toolbar, or by choosing **Program > Do**. To change the program later, type `MODIFY COMMAND hello` again.

What happens when something goes wrong

Mistakes are part of programming, so let us make one on purpose. Open the program again and change the second line so that it is missing its closing quote:

```
? "Hello from Ravensby!
```

Save it and run `DO hello`. VFP cannot compile the line, so it stops and shows a dialog. You should see something like this:

```
+-- Program Error ------------------------------------+
|  Command contains unrecognized phrase/keyword.      |
|                                                     |
|  [Cancel]  [Suspend]  [Ignore]  [Help]              |
+-----------------------------------------------------+
```

The exact wording depends on the mistake. The buttons are worth knowing: **Cancel** stops the program and returns you to the Command Window; **Suspend** pauses it so you can inspect variables (useful later, with the Debugger); **Ignore** skips the failing line and carries on, which is rarely what you want. Choose Cancel, fix the quote, save, and run again. Chapter 27 covers error handling in depth, including how a finished application should report errors to its users instead of showing this developer dialog.

> **WARNING**
>
> Avoid naming your programs after VFP commands or functions, such as `date.prg`, `list.prg`, or `select.prg`. VFP may run its own command instead of your file, or run your file when you meant the command, and the confusion can take a long time to untangle. Names such as `hello.prg`, `makedata.prg`, and `loanlib.prg` are safe and descriptive.

Files You Will Meet

VFP uses many file extensions, and it helps to recognize them early. You do not need to remember this table; come back to it whenever an unfamiliar extension appears in a folder.

Extension	What it is	First used in
`.prg` / `.fxp`	Program source / compiled program	Chapter 1
`.dbf`	Table (data records)	Chapter 3
`.fpt`	Memo data for a table (long text)	Chapter 3
`.cdx`	Compound index for a table	Chapter 7
`.dbc` / `.dct` / `.dcx`	Database container and its memo and index files	Chapter 10
`.pjx` / `.pjt`	Project file and its memo file	Chapter 5
`.scx` / `.sct`	Form and its memo file	Chapter 9
`.frx` / `.frt`	Report and its memo file	Chapter 9
`.vcx` / `.vct`	Visual class library and its memo file	Chapter 17
`.mnx` / `.mnt` / `.mpr`	Menu, its memo file, generated menu program	Chapter 25
`.exe`	Finished application	Chapter 5

Notice the pattern: many designer files come in pairs, such as `.scx` and `.sct` for a form. Both halves belong together; when you copy or back up a form, always copy both files, or the form will not open.

How to Use This Book

A few conventions make the chapters easier to follow.

- **Code captions.** A gray caption above a code block tells you where the code goes. "Command Window" means type it there, line by line. A file path such as `C:\Ravensby\QuickStart\hello.prg` means the code is the content of that file. "Output" means what VFP printed when we ran the code.
- **Real output.** Every output block in this book was produced by running the code with VFP 9 using US date and currency settings. When something cannot be run automatically (a dialog that waits for you to click, a designer window) we say "you should see something like" and show a sketch.
- **Today is Monday 03/16/2026.** Functions in the projects take explicit dates rather than calling `DATE()` inside their logic. That way the fines, fees, and bookings in the examples come out exactly the same on your PC as on ours, whatever the real date is. The screens you build pass `DATE()` when a real user works with them.
- **Naming.** Variables start with a scope letter and a type letter: `lcName` is a local character variable, `lnTotal` a local number, `ldDate` a local date. Chapter 2 explains the full scheme. VFP keywords are written in UPPERCASE so you can tell them apart from names you invent.
- **Sample data scripts.** Every project has a program called `progs\makedata.prg` that creates its tables and fills them with sample rows. You can run it again at any time to get back to a clean starting point, so feel free to experiment and break things.
- **Practice and Key Takeaways.** Each chapter ends with exercises (with hints) and a short list of the most important points. If you are short on time, read the Key Takeaways first and come back for the details.

Tips, notes, and warnings appear in shaded boxes. Tips are shortcuts and good habits, notes add background, and warnings point out traps that cost real developers real time. Each warning in this book describes behavior we verified ourselves.

If you know Access or Excel

From Microsoft Access, tables, queries, forms, and reports will feel familiar; the differences are that the language sits at the center, tables are separate files, and the Command Window drives everything. From Excel, think of a table as a sheet where every column has a fixed type and width and formulas live in programs, not cells.

Common Mistakes in the First Hour

- **Forgetting the current folder.** If `DO hello` reports that the file does not exist, check where you are with `? SYS(5) + CURDIR()` and use `CD C:\Ravensby\QuickStart`.
- **Typing into the wrong window.** If your keystrokes seem to vanish, you may be typing into the program editor or another window. Click in the Command Window or press Ctrl+F2.
- **Mismatched quotes.** A string must start and end with the same kind of quote. VFP accepts double quotes, single quotes, and square brackets: `"text"`, `'text'`, and `[text]`.