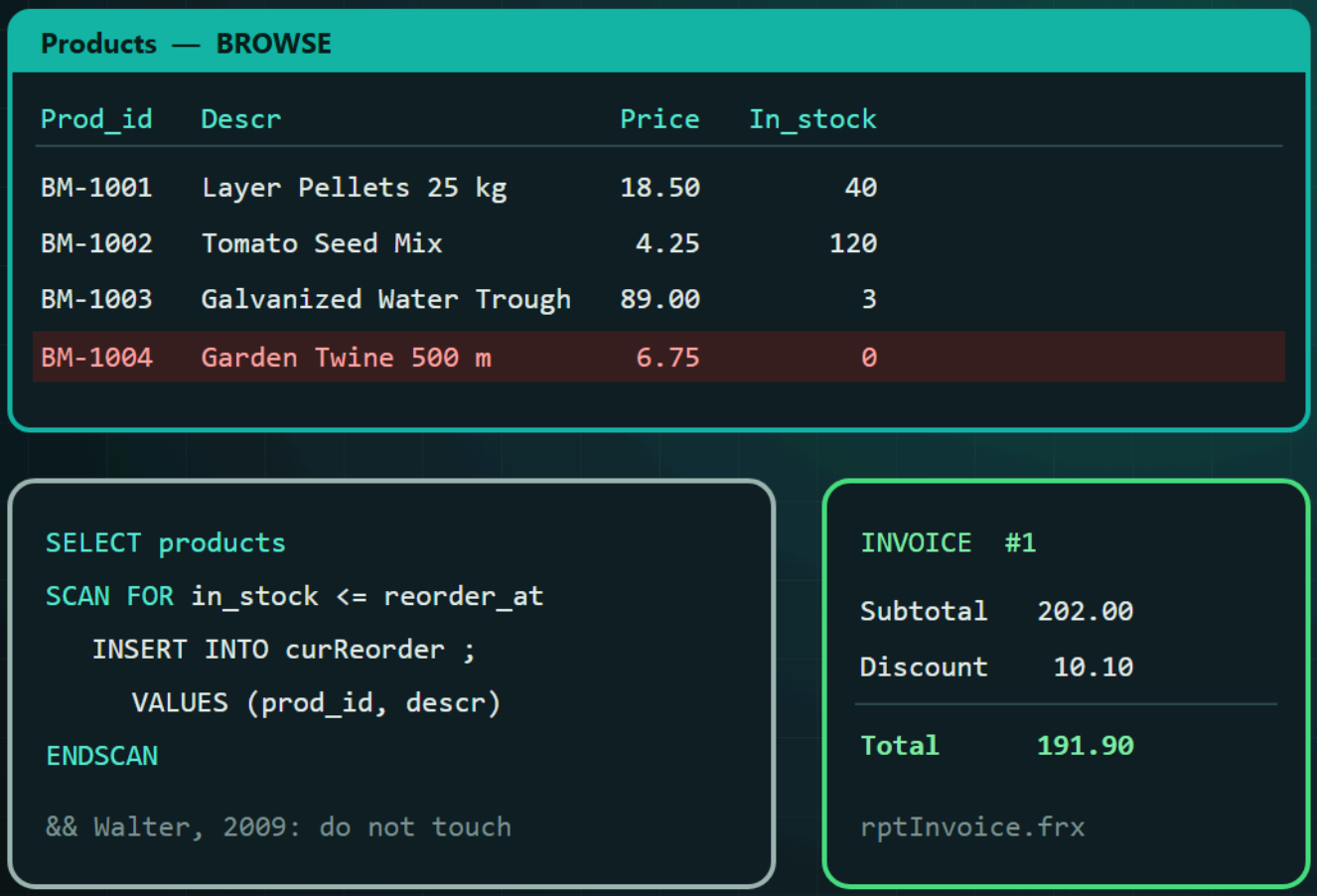


TABLES · SQL · FORMS · REPORTS · SQL SERVER · MIGRATION

MAINTAIN · EXTEND · MODERNIZE

Visual FoxPro in Practice

A complete, practical guide to legacy VFP 9 applications

From tables and Xbase to forms, reports, SQL Server, and a safe path to modernization

DBC · RUSHMORE · CURSORADAPTER · COM · JSON

THENEXBOOKS

Visual FoxPro in Practice

Maintain, Extend, and Modernize Legacy VFP 9 Applications

First Edition

THENEXBOOKS

Visual FoxPro in Practice
Maintain, Extend, and Modernize Legacy VFP 9 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. 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. Always back up data before running any command that changes or repairs tables.

Microsoft, Visual FoxPro, Windows, Excel, Word, and SQL Server are trademarks of the Microsoft group of companies. Their use in this book is for identification only and does not imply endorsement by, or affiliation with, Microsoft. Other product names are trademarks of their respective owners. This book does not include or distribute Visual FoxPro; you need your own licensed copy.

Bellmoor Farm Supply, its staff, customers, and products are fictional. Any resemblance to real companies or people is coincidental.

Preface

Somewhere in your organization, or in the organization that just hired you, there may be a Visual FoxPro application that has quietly done its job for fifteen or twenty-five years. It takes orders, prints invoices, tracks stock, or runs payroll. The people who use it every day know exactly how it behaves. The person who wrote it may have retired. And now it is your job to keep it running, add the feature the business needs this quarter, and help decide what happens to it next.

This book is written for that job. Visual FoxPro is no longer developed or sold by Microsoft, so almost nobody starts a brand-new FoxPro project today — but a great deal of working software was built with it, and it was built that way for good reasons. VFP combines a fast local database engine, a capable SQL dialect, an object-oriented language, and visual designers for forms and reports in one tool. Once you understand how those pieces fit together, legacy code that looked mysterious becomes readable, and changes that looked risky become routine.

We start from zero: the development environment, tables and data types, and the language itself. Then we cover everything a maintainer meets in real applications — work areas and record pointers, indexes and Rushmore, database containers and triggers, SQL, buffering and transactions, classes and visual class libraries, forms, grids, menus, reports, error handling, deployment, and multi-user locking. Finally we look outward: SQL Server and ODBC, COM and the Windows API, XML and JSON, performance and data integrity, and practical, low-risk paths to modernize or migrate.

Who This Book Is For

- **Developers who have inherited a VFP application** and come from C#, VB, Java, Python, or SQL, and need to become productive in FoxPro quickly.
- **Junior developers** at companies that still run business systems on Visual FoxPro.
- **Experienced FoxPro developers** who want a structured refresher on VFP 9 practices, client/server work, and modernization.
- **Technical leads and IT managers** planning the future of a VFP system who want to understand what they are dealing with.

What You Will Learn

- The VFP 9 environment, tables and data types, and the Xbase language: variables, expressions, functions, control flow, and procedures.
- Working with data: work areas, record navigation, indexes, Rushmore optimization, database containers, rules, and triggers.
- SQL in Visual FoxPro, cursors, buffering, and transactions.
- Object-oriented VFP: classes, visual class libraries, and a framework layer for forms and controls.

- Forms, data environments, private data sessions, grids, combo boxes, menus, and the application shell.
- Reports and report listeners, error handling and debugging, files and text output, and building and deploying an EXE.
- Client/server with SQL Server, COM automation and the Windows API, XML, JSON, and web services.
- Performance, data integrity, multi-user locking, migration planning, and maintaining a legacy codebase safely.

How This Book Is Organized

Part I — Getting Started (Chapters 1–5) introduces the environment, tables, and the language. **Part II — Working with Data** (Chapters 6–10) covers navigation, indexes, databases, SQL, buffering, and transactions. **Part III — Object-Oriented Visual FoxPro** (Chapters 11–15) covers classes, class libraries, forms, data sessions, and data-heavy controls. **Part IV — Building Applications** (Chapters 16–20) covers menus, reports, error handling, files, and deployment. **Part V — Integration and Modernization** (Chapters 21–25) covers SQL Server, COM, XML and JSON, performance and integrity, and multi-user applications. **Part VI — Projects, Migration, and Next Steps** (Chapters 26–28) walks through the complete application, plans a migration, and shows how to maintain a legacy codebase with confidence. The appendices provide a command and function reference, a guide to SET commands, an error messages field guide, and a glossary.

Each chapter includes practice exercises and ends with a **Key Takeaways** box for quick review.

The Running Example: Bellmoor Farm Supply

Throughout the book we work on **BELLORD**, the order-entry system of **Bellmoor Farm Supply**, a regional farm-and-garden supplier. Walter wrote BELLORD years ago and has since retired, leaving behind working code and a few cryptic comments. Ruth owns the business, Marta runs the order counter and reports problems with admirable precision, and Sam — the newly hired developer, and your stand-in — has inherited the system. Chapter by chapter we rebuild and understand BELLORD: its tables and database, its order logic and discount rule, its forms, menus, and reports, and finally its connection to SQL Server and its path to the future.

Versions and Conventions

Everything in this book targets Visual FoxPro 9.0 with Service Pack 2. Most of it also applies to VFP 8 and earlier versions; features that need VFP 9 are pointed out. Keywords are written in uppercase and variables use the Hungarian-style prefixes common in FoxPro code (`lcName`, `lnTotal`, `tyAmount`). Third-party and community tools are mentioned as options; check their own documentation for current details.

> **TIP**
>
> A recommended practice or a technique that saves you time.

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

> **WARNING**
>
> A common mistake, a destructive command, or a habit to avoid.

File captions such as *C:\Bellmoor\Prog\orderbiz.prg* tell you where a snippet belongs.

Contents

Preface

PART I — GETTING STARTED

1 Visual FoxPro Today: What It Is, Why It Still Matters, and Who This Book Is For

2 The VFP 9 Environment: Command Window, Project Manager, and Your First Program

3 Tables, Fields, and Data Types: Creating and Browsing DBF Tables

4 Language Basics: Variables, Expressions, Operators, and Built-In Functions

5 Control Flow, Procedures, Functions, and Parameters

PART II — WORKING WITH DATA

6 Navigating and Changing Data: Work Areas, Records, and Xbase Commands

7 Indexes and Fast Lookups: CDX Tags, SEEK, and Rushmore

8 Databases, Relations, Rules, and Triggers

9 SQL in Visual FoxPro: SELECT, INSERT, UPDATE, DELETE, and Cursors

10 Buffering and Transactions

PART III — OBJECT-ORIENTED VISUAL FOXPRO

11 Classes and Objects: DEFINE CLASS, Properties, Methods, and Inheritance

12 Visual Class Libraries and Designing a Framework Layer

13 Forms: The Form Designer, Controls, Events, and Data Binding

14 Data Environments, Private Data Sessions, and Multi-Form Applications

15 Grids, Combo Boxes, and List Boxes

PART IV — BUILDING APPLICATIONS

16 Menus, Toolbars, and the Application Shell

17 Reports and Labels: The Report Designer and Report Listeners

18 Error Handling and Debugging

19 Strings, Dates, Files, and Text Output

20 Building, Distributing, and Deploying Applications

PART V — INTEGRATION AND MODERNIZATION

21 Client/Server: SQL Server, ODBC, SQL Pass-Through, Remote Views, and CursorAdapter

22 COM, Automation, and the Windows API

23 XML, JSON, and Web Services

24 Performance, Data Integrity, and Maintenance

25 Multi-User Applications: Locking, Buffering Strategies, and Conflicts

PART VI — PROJECTS, MIGRATION, AND NEXT STEPS

26 Project: The BELLORD Order Entry Application End to End

27 Migrating Data and Planning a Modernization

28 Maintaining a Legacy Codebase and Where to Go Next

APPENDICES

A Commands and Functions Quick Reference

B SET Commands and Environment Settings

C Error Messages Field Guide

D Glossary

PART I

Getting Started

What Visual FoxPro is and why so many business systems still depend on it, the development environment, tables and data types, and the Xbase language you will read in every VFP application.

1 · Visual FoxPro Today
2 · The VFP 9 Environment
3 · Tables and Data Types
4 · Language Basics
5 · Control Flow and Procedures

CHAPTER 1 · PART I — GETTING STARTED

Visual FoxPro Today: What It Is, Why It Still Matters, and Who This Book Is For

Visual FoxPro has not had a new release in almost two decades, yet the applications built with it still take orders, print invoices, and keep stock counts honest every working day. This chapter explains what VFP is, where it stands today, what kinds of systems you are likely to inherit, and how this book will turn you into the person who can keep one of them running — and eventually move it forward.

A Monday Morning at Bellmoor Farm Supply

Bellmoor Farm Supply is a regional wholesaler of farm and garden goods: layer pellets by the 25 kg sack, seed mixes, galvanized water troughs, garden twine, and a few thousand other things that orchards, stables, and market gardeners buy in bulk. Customers phone in or walk up to the order counter, a clerk keys the order, the warehouse picks it, and an invoice goes out with the truck.

All of that runs on one program called **BELLORD**, short for "Bellmoor Orders". It was written in Visual FoxPro by **Walter**, who built the first version many years ago and kept improving it until he retired. Walter was a careful developer in most places and a hurried one in a few. He left comments throughout the code — some are genuinely helpful ("discount is on the subtotal, not per line — Ruth's rule"), and some are cryptic ("don't touch, see the March thing").

Ruth owns the business. She does not care what language BELLORD is written in; she cares that orders go out, that customers are not given more credit than they have earned, and that the reorder report tells her what to buy before the shelves are empty. **Marta** runs the order counter. She is the person who notices when something is wrong, and she reports problems precisely: which screen, which customer, which button, what the message said, and what she expected instead. Every developer should be lucky enough to have a Marta.

And then there is **Sam**, the newly hired developer. Sam has a solid background in C# and SQL, has written web applications, and has never opened Visual FoxPro in his life. On his first Monday, Ruth gave him a desk, a login, and a sentence that sums up this entire book: "BELLORD works, mostly. Keep it working, make it better, and tell me honestly what we should do with it in the long run."

Sam is your stand-in. Whatever brought you here — an inherited system at work, a contract to maintain someone else's application, a migration project, or a wish to refresh skills you last used years ago — you are in Sam's chair. Throughout the book we follow him as he learns the platform, reads Walter's code, fixes the problems Marta reports, adds features Ruth asks for, and prepares BELLORD for whatever comes next.

What Visual FoxPro Actually Is

Visual FoxPro (VFP) is three things packaged together, and newcomers often only notice one of them.

- **A programming language.** VFP's language descends from the xBase family (dBASE, FoxBASE, Clipper). It mixes record-oriented commands such as `USE`, `SKIP`, and `REPLACE` with a full SQL dialect (`SELECT`, `INSERT`, `UPDATE`, `DELETE`) and, since version 3.0, a complete object-oriented model with classes, inheritance, and events.
- **A database engine.** VFP reads and writes its own table format — the `.dbf` file — directly, with no separate server process. Tables can live alone ("free tables") or be grouped in a database container (`.dbc`) that adds long field names, validation rules, triggers, stored procedures, and persistent relations. The engine includes Rushmore, an optimization technology that uses indexes to make filtered queries fast.
- **A development environment.** The VFP IDE contains a Command window for running code interactively, designers for forms, classes, menus, reports, and tables, a Project Manager that ties everything together, a debugger, and a compiler that builds a Windows executable.

The combination is what made VFP so productive: design a table, query it in the Command window, show the result in a grid on a form, and compile it all into an `.exe` — without switching tools or administering a database server.

> **NOTE**
>
> If you come from C# and SQL Server, a useful mental model is "a language in which the data access layer, the database engine, and the UI framework are all built in and share one runtime". In VFP, a table that is open is directly visible to the code: you can write `? products.price` and get the price of the current product row, with no connection, command, or reader object in between. That directness is both VFP's greatest strength and the source of many maintenance headaches, as later chapters show.

A short history

The FoxPro line began at Fox Software, an Ohio company whose FoxBASE product was a fast dBASE-compatible database system. FoxPro followed, and FoxPro 2.0 introduced Rushmore query optimization — the reason "Fox is fast" became a saying among database developers. Microsoft acquired Fox Software in 1992 and continued the product, first as FoxPro 2.x for DOS and Windows, and then, in 1995, as **Visual FoxPro 3.0**, which added object orientation, visual designers, and the database container.

Several versions followed, each adding significant features. VFP 8 brought structured error handling with `TRY...CATCH...FINALLY`, the `CursorAdapter` class, and auto-incrementing integer fields. **Visual FoxPro 9.0 was released in December 2004**. It added the Varchar, Varbinary, and Blob data

types, a much more standards-compliant SQL engine, report listeners that opened up the reporting system, and many smaller improvements. **Service Pack 2 for VFP 9 followed in 2007**, and it is the version this book targets: Visual FoxPro 9.0 SP2.

Where VFP stands today

We will be direct about the platform's status, because you will need to explain it to people like Ruth.

- In **March 2007, Microsoft announced that there would be no Visual FoxPro 10**. VFP 9 SP2 is the last version.
- **Extended support ended in January 2015.** Microsoft no longer issues fixes or security updates for VFP.
- **Microsoft no longer sells Visual FoxPro.** To develop with it, you need an existing licensed copy — typically from an older MSDN subscription or from installation media the company bought years ago. If your employer runs a VFP application, ask whoever handles software assets where the original media and license records are; they are usually in a cupboard or on a file share somewhere.
- The VFP **community remains active**. The main community project hub is **VFPX**, hosted on GitHub, which collects open-source tools, libraries, and extensions — several of which we use later in the book.

> **WARNING**
>
> Do not download Visual FoxPro installers from unofficial websites. Such files are unlicensed, and an executable from an unknown source is a security risk on a machine that has access to company data. Use only the licensed media your organization owns. If you cannot find a licensed copy, raise that with management before you start — it is a business decision, not a technical workaround.

Applications built with VFP 9 are 32-bit Windows programs, and in practice they generally continue to run on current versions of Windows. "Generally" matters, though: because Microsoft no longer tests or supports the runtime, each new Windows release, network configuration, or antivirus product is something your own team must test. Part of maintaining a VFP application responsibly is knowing that the ground under it is no longer being maintained by its vendor, and planning accordingly. Chapter 24 and Chapter 27 return to this in detail.

Why VFP Still Matters

If the product is finished, why learn it? Because the applications are not finished. Businesses do not replace software because a vendor stops selling a compiler; they replace it when the software stops doing its job or when the risk of keeping it becomes larger than the cost of replacing it. Many VFP

applications are still doing their jobs well. They encode years of business rules — discount policies, credit checks, pricing exceptions, odd tax treatments, reports formatted exactly the way an accountant asked for in 2009 — that are documented nowhere except in the code.

That last point is the important one. A rewrite that starts by ignoring the old system usually rediscovers its business rules one production incident at a time. A rewrite (or a gradual modernization) that starts with someone who can actually read the old code, run it, and ask "why does it do this?" goes much better. Whether Bellmoor keeps BELLORD for ten more years or replaces it within two, Sam needs to understand it first.

There are also solid technical reasons VFP applications have lasted:

- **Fast local data handling.** The engine processes tables of hundreds of thousands of rows comfortably on a desktop machine, and Rushmore-optimized queries return quickly when the indexes are right (Chapter 7 and Chapter 24).
- **Low infrastructure.** A small company can run a multi-user VFP application with nothing more than a shared network folder. No database server, no web server.
- **Strong integration options.** VFP can talk to SQL Server and other databases through ODBC (Chapter 21), automate Excel and Word through COM, call Windows API functions (Chapter 22), and work with XML and web services (Chapter 23). Many VFP applications quietly became front ends to SQL Server databases years ago.

We will be equally honest about the limits: a 2 GB size limit per table file, a file-sharing model that is sensitive to network problems, a 32-bit runtime, no vendor support, and a shrinking pool of developers. None of these is a reason to panic. All of them are reasons to have a plan.

What Kinds of VFP Applications You Will Meet

Most VFP code in the wild falls into a few recognizable shapes. Knowing which one you have inherited tells you where to look first.

Kind of application	Typical features	Where to look first
Desktop line-of-business system	Forms, menus, reports, DBF tables on a shared folder	Main program, menu, data folder
Front end to SQL Server	Remote views, SQL pass-through, CursorAdapters	Connection code, views in the DBC
Batch or utility program	No forms; imports, exports, nightly jobs	The scheduled command line
COM server	OLEPUBLIC classes called by other programs	Class definitions, registration
Data tool for analysts	Command-window scripts, ad hoc queries	Loose `.prg` files

BELLORD is the first kind: a classic desktop line-of-business system with DBF tables in a shared folder. It is the most common shape and the best one to learn on, because every other shape uses the same language, the same data commands, and the same debugging tools.

The Anatomy of a VFP Application on Disk

When Sam first opened the BELLORD folder, he saw dozens of file extensions he did not recognize. VFP stores most of its designers' output as tables — a form is really a table of controls and code — and each designer uses a pair of files: one with the structure and one with memo data. Here is the field guide.

Extension	What it holds	Covered in
`.prg` / `.fxp`	Program source / compiled program	Chapters 2, 5
`.dbf` / `.fpt`	Table data / memo, General, Blob data	Chapter 3
`.cdx`	Compound index (index tags)	Chapter 7
`.dbc` / `.dct` / `.dcx`	Database container and its memo and index	Chapter 8
`.vcx` / `.vct`	Visual class library	Chapter 12
`.scx` / `.sct`	Form	Chapter 13
`.mnx` / `.mnt`	Menu definition	Chapter 16
`.mpr` / `.mpx`	Generated menu program / compiled	Chapter 16
`.frx` / `.frt`	Report	Chapter 17
`.pjx` / `.pjt`	Project	Chapters 2, 20
`.exe` / `.app`	Built application	Chapter 20
`config.fpw`	Startup settings (plain text)	Chapters 2, 20

> **TIP**
>
> Always treat the paired files as one unit. A form is the `.scx` *and* its `.sct`; a table is the `.dbf` *and* its `.fpt` and `.cdx`. Copying or restoring only half of a pair is one of the most common ways a working VFP application gets broken during a "quick fix". Because designer files are binary tables, they also do not diff well in source control; Chapter 28 shows how the VFPX tool FoxBin2Prg solves that.

What BELLORD Does

Before writing a line of code, Sam spent his first afternoon at the order counter watching Marta work. That is the best way to learn any business system, and it gave him the functional map of BELLORD that the rest of this book builds, piece by piece.

The data

BELLORD keeps four tables in a database container called `bellmoor.dbc`, stored in `C:\Bellmoor\Data\`:

- `customers` — who buys from Bellmoor: an ID such as `C00001`, a name, a city, a phone number, a credit limit, and whether the account is active.
- `products` — what Bellmoor sells: an ID such as `BM-1001`, a description, a category (FEED, SEED, EQUIP, GARDEN), a price, the quantity in stock, and the reorder level.
- `orders` — one row per order: an auto-incrementing order number, the customer, the date, a status (`O` open, `S` shipped, `C` cancelled), and the subtotal, discount, and total.
- `orditems` — the lines of each order: order number, line number, product, quantity, and the unit price at the time of sale.

At the start of our story, the product list looks like this:

prod_id	descr	category	price	in_stock	reorder_at
BM-1001	Layer Pellets 25 kg	FEED	18.50	40	10
BM-1002	Tomato Seed Mix	SEED	4.25	120	30
BM-1003	Galvanized Water Trough	EQUIP	89.00	3	2
BM-1004	Garden Twine 500 m	GARDEN	6.75	0	15

And there are three customers: **C00001** Ridgeview Orchard in Ashford (credit limit 1,500.00), **C00002** Hannah Pryce, also in Ashford (credit limit 300.00), and **C00003** Copper Creek Stables in Dunmore (credit limit 2,500.00). All three are active. (Every business and person in BELLORD's data is fictional.)

The business rules

Ruth explained the rules to Sam in five minutes. They took Walter considerably longer to implement, and they will take us most of the book to rebuild properly:

1. **Money is exact.** Prices and totals are stored using VFP's Currency type, never floating-point numbers, and displayed with a dollar sign and two decimals, such as `$18.50`.
2. **Volume discount.** When an order's subtotal is 200.00 or more, the customer gets 5% off the subtotal, rounded to two decimals. The total is the subtotal minus the discount. BELLORD does not calculate tax.

3. **Stock is checked line by line.** A line can only be added when the quantity ordered is no more than the quantity in stock. Adding a line reduces the stock; cancelling an open order puts it back.
4. **Credit is checked per order.** An order is refused when the customer's open orders plus the new order would exceed the customer's credit limit.
5. **Reorder report.** Every product whose stock is at or below its reorder level appears on the reorder report.

Ruth also gave Sam two examples, which we use as test cases throughout the book. The first is **order #1**: on 2 March 2026, Ridgeview Orchard (C00001) ordered 10 sacks of Layer Pellets (10 × 18.50 = 185.00) and 4 packets of Tomato Seed Mix (4 × 4.25 = 17.00). The subtotal is **202.00**, which qualifies for the discount: 5% of 202.00 is **10.10**, so the total is **191.90**. Afterwards, Layer Pellets stock drops from 40 to 30 and Tomato Seed Mix from 120 to 116.

The second is **order #2**: Hannah Pryce (C00002) asks for 2 rolls of Garden Twine. The order is refused, because there are 0 in stock. BELLORD should say so clearly — Marta's preferred wording is "BM-1004 is out of stock".

As for the reorder report: at the start, only BM-1004 (Garden Twine, 0 in stock against a reorder level of 15) is listed. The water trough is *not* listed: 3 in stock is still above its reorder level of 2. After order #1, the list is unchanged, since 30 pellets is above 10 and 116 seed packets is above 30.

A first look at the code

Here is a taste of what VFP code looks like — a short fragment in the style Sam found in Walter's programs. Do not worry about the details; every command here is explained in later chapters (Chapter 4 for the expressions, Chapter 6 for `SCAN`, Chapter 9 for the SQL).

A Walter-style fragment (for reading, not typing yet)

```
* Reorder check -- WK
* products where stock is at or below reorder level
SELECT prod_id, descr, in_stock, reorder_at ;
   FROM products ;
   WHERE in_stock <= reorder_at ;
   INTO CURSOR curReorder

SELECT curReorder
SCAN
   ? prod_id, descr, in_stock   && prints BM-1004 at the start
ENDSCAN
USE IN curReorder              && close the cursor again
```

Even without knowing VFP, you can probably read most of it. A SQL query selects low-stock products into a temporary result called a *cursor*; a `SCAN` loop walks through each row; `?` prints values to the screen. The semicolon at the end of a line continues a command onto the next line, `*` at the start of a

line marks a comment, and `&&` starts a comment at the end of a line. That blend of SQL and record-by-record commands is typical of real VFP code, and learning to move comfortably between the two styles is one of the main goals of Part II.

Reading the Error: Marta's First Bug Report

On Sam's second day, Marta sent him this note: "Tried to open BELLORD on the new counter PC. Got a box saying the file does not exist, with an OK button. Old PC works fine. Nothing else happened." She attached a photo of the screen:

```
+------------------------------------------------+
| Microsoft Visual FoxPro                   [X] |
+------------------------------------------------+
|  (!)  File 'bellmoor.dbc' does not exist.      |
|                                                |
|                       [  OK  ]                 |
+------------------------------------------------+
```

This is VFP **error 1, "File does not exist"**, one of the most common errors you will see. Reading it carefully tells you a lot:

- The message names the file VFP could not find: `bellmoor.dbc`, the database container. So the program itself started — the executable and its runtime were found — and the failure happened when the code tried to open the data.
- The file name has no folder in front of it. That means the code opens the database with a relative name and relies on the *default directory* or the *search path* to find it. On the old PC, those settings point at the right folder; on the new one, they do not.
- "Old PC works fine" rules out a missing or damaged database — the file obviously still exists on the share.

The fix turned out to be a missing `config.fpw` file on the new PC, the small text file that sets the default directory and path when the application starts. Chapter 2 explains `SET DEFAULT`, `SET PATH`, and `config.fpw`, and Chapter 20 shows how to deploy them so this cannot happen again. For now, the lesson is the habit: read the exact message, note which file or variable it names, and ask "what is different between the case that works and the case that fails?"

Who This Book Is For

We wrote this book for three groups of readers:

- **Developers who have inherited a VFP application** and come from another background — C#, VB, Java, Python, or SQL. You know how to program; you need to know how VFP does things, and why the code you inherited looks the way it does. Look for the short "if you come from..." asides.

- **Junior developers at companies that still run VFP systems.** You may be learning VFP as one of your first platforms. Every concept is explained from scratch, and every example is something you can type and run.
- **Longtime FoxPro developers** who want a structured refresher on VFP 9 practices — buffering, `TRY...CATCH`, report listeners, CursorAdapter — and a practical approach to modernization and migration.

You will need a Windows PC and a licensed installation of Visual FoxPro 9.0, ideally with Service Pack 2 applied. You can check what you have by typing this in the Command window (Chapter 2 introduces it properly):

```
? VERSION()
```

On VFP 9 the result starts with `Visual FoxPro 09.00.0000` followed by a build number. Compare that build with the one documented for your installation media (SP2 and later hotfixes report higher builds than the original release); if it is older, ask whether the service pack is available among your organization's licensed media.

The Plan for the Book

The book follows Sam's path from "never seen VFP" to "owns BELLORD". Each part builds on the previous one, and the BELLORD application grows chapter by chapter.

Part I — Getting Started (Chapters 1–5)

Chapter 2 tours the VFP 9 environment — the Command window, Project Manager, Data Session window, Properties window, and a first look at the Debugger — and sets up the `C:\Bellmoor\` folders. Chapter 3 creates BELLORD's four tables as free tables and loads the canonical products and customers. Chapter 4 covers the language basics: variables, data types, operators, and built-in functions. Chapter 5 adds control flow, procedures, and parameters, and writes the first shared functions, `FormatMoney()` and `CalcDiscount()`, into the procedure file `bmlib.prg`.

Part II — Working with Data (Chapters 6–10)

Chapter 6 teaches the Xbase commands that move through and change records. Chapter 7 adds indexes and the fast lookups they enable. Chapter 8 moves the free tables into the database container `bellmoor.dbc`, with primary keys, relations, rules, and triggers. Chapter 9 covers VFP's SQL dialect, including the reorder query. Chapter 10 introduces buffering and transactions and places order #1 safely inside a transaction.

Part III — Object-Oriented Visual FoxPro (Chapters 11–15)

Chapter 11 builds the `OrderManager` business object that enforces stock, credit, and discount rules. Chapter 12 creates the visual class library `bmbase.vcx`. Chapters 13 to 15 build the customer, product, and order forms, with data environments, private data sessions, grids, and combo boxes.

Part IV — Building Applications (Chapters 16–20)

Chapter 16 assembles the menu and the main program into an application shell. Chapter 17 builds the invoice and reorder reports. Chapter 18 covers error handling and debugging, including a global error handler. Chapter 19 handles strings, dates, files, and text output, and Chapter 20 builds and deploys `bellord.exe`.

Part V — Integration and Modernization (Chapters 21–25)

Chapter 21 connects VFP to SQL Server. Chapter 22 covers COM automation and the Windows API. Chapter 23 works with XML, JSON, and web services. Chapter 24 addresses performance and data integrity, and Chapter 25 covers multi-user locking and update conflicts.

Part VI — Projects, Migration, and Next Steps (Chapters 26–28)

Chapter 26 walks through BELLORD end to end. Chapter 27 plans a migration of the data and, gradually, the application. Chapter 28 is about the craft of maintaining a legacy codebase: reading unfamiliar code, source control, safe refactoring, testing, and the community resources that will keep helping you after this book.

How to Read the Code in This Book

A few conventions make the examples easier to follow:

- A caption above a code block, such as `C:\Bellmoor\Prog\bmlib.prg`, tells you which file the code belongs in. Code without a caption is meant to be typed into the Command window, one line at a time.
- VFP keywords are in UPPERCASE (`SELECT`, `SCAN`, `LPARAMETERS`). VFP itself is not case-sensitive about keywords or names, but the convention makes code easier to scan.
- Variables use Hungarian-style prefixes, as Walter's code does: the first letter is the scope (`l` for local, `t` for a parameter) and the second is the type (`c` character, `n` numeric, `y` currency, `d` date, `l` logical, `o` object, `a` array). So `lyTotal` is a local Currency variable, and `tcCustId` is a character parameter. Temporary result sets (cursors) start with `cur`, as in `curOpenOrders`.
- When a command prints something, we show the output in a comment or in a separate block, so you can compare it with what you see.

VFP lets you abbreviate most commands and keywords to their first four letters — `REPL` for `REPLACE`, `MODI COMM` for `MODIFY COMMAND`. You will see this in older code, including Walter's. We always write the full words, because they are far easier to read and search for.

Common Mistakes

- **Deciding to rewrite before understanding.** A rewrite that begins without reading the existing code loses business rules. Learn the system first; Chapter 27 shows how to plan modernization step by step.
- **Treating DBF data as unimportant because the format is old.** The tables hold the company's live orders, customers, and stock. Back them up before any experiment, and never test against the production share.
- **Searching the web for an installer.** Use your organization's licensed media only. Unofficial downloads are a licensing and a security problem.
- **Copying half of a file pair.** Moving a `.scx` without its `.sct`, or a `.dbf` without its `.fpt` and `.cdx`, produces errors that look mysterious until you remember the pairs.
- **Mocking the original developer's code.** Walter worked under constraints Sam cannot see. Odd code often handles an odd business case. Find out why before you "clean it up".

Practice

1. **Inventory your installation.** Find out which VFP version and build is installed on your development machine (or will be), and where the licensed media is kept. *Solution:* in the Command window, `? VERSION()`, and compare the build number with the service-pack notes for your licensed media. Record the location of the media and license information in your team's documentation.
2. **Classify an application.** Using the table of application kinds above, classify a VFP application you work with (or BELLORD). What are the first three files you would open? *Solution:* BELLORD is a desktop line-of-business system; open `main.prg`, the menu `bellmoor.mnx`, and the database `bellmoor.dbc`.
3. **Check the discount by hand.** A customer orders 8 × BM-1003. What are the subtotal, discount, and total? *Solution:* 8 × 89.00 = 712.00; that is at least 200.00, so the discount is ROUND(712.00 × 0.05, 2) = 35.60 and the total is 676.40. (In real life the order would be refused anyway: only 3 troughs are in stock.)
4. **Predict the reorder report.** If a customer bought 1 water trough (BM-1003), would the reorder report change? *Solution:* yes — stock would drop from 3 to 2, and 2 <= 2, so BM-1003 would join BM-1004 on the list.
5. **Read an error.** A user reports "File 'products.dbf' does not exist." What two settings would you check first? *Solution:* the default directory and the search path (`SET DEFAULT` and `SET PATH`), usually set in `config.fpw` or the main program.