Contents

Acknowledgments iv

Introduction **1**
A Brief Description of Our Journey 1
From Old Problematic Monoliths to Innovative, Lightweight, Efficient Programs 3
"Why ILE? OPM Has Served Me Fine So Far" 4
The Virtues of ILE, by IBM Itself (with a Little Help from Me) 4
Modularity, or Playing with Legos 5
Reusable Components—Don't Rewrite; Reuse! 5
Common Runtime Services—Don't Reinvent the Wheel 6
Source Debugger—No Longer the ISDB Nightmare 6
Summary 6

PART 1: ILE BASICS **7**

Chapter 1: Modules, Programs, and Service Programs **9**
A Typical OPM Scenario 10
A Basic ILE Approach to This Problem 11
Why Some Detractors Say ILE Means "It's a Link Editor" 13
Anatomy of a Module Object 14

An Even Better Way: In Comes the Service Program 15
Summary 16

Chapter 2: Binding It All Together 19
Before We Begin, Let's Review the Journey 19
Start Here: Compiling Your First Module 21
Next Stop: Service Programs (with a Detour to Explain Binding) 21
Our Final Destination: Compiling a Program Object 24
No Trip Is Perfect Without a Couple of Unexpected Situations 25
... And a Few Shortcuts 26
Summary 26

Chapter 3: Procedures: How, When, and Why to Build Them 29
How to Build Your First Procedure, Using Your OPM Knowledge 30
When and Why You Should Build a Procedure 35
A Side Note: /COPY or /INCLUDE? 37
Summary 39

Chapter 4: Improve Your Code's Readability with Functions 41
Reshaping the "Check If Item Exists in Inventory" Procedure into a Function 42
The Importance of Being a Function, or the Return Value Concept 45
Functions Within Functions Made Simple 47
Nested Functions: A Different Approach to Functions Within Functions 51
Summary 53

Chapter 5: All About Parameters 55
Adding VALUE to Your Parameters 56
Keeping It CONSTant 57
Value Versus CONST 58
So, What Keyword Should I Use for My Input Parameters? 59
You Need to Know Your OPTIONS 60
Allowing (and Handling) Missing-in-Action Parameters with *NOPASS 60
A Different Approach to the MIA Parameter Problem 64
*NOPASS Versus *OMIT 67

The Morphing Parameter (Wait, What?!) 67
C-like Strings 69
(Sort of) Self-Formatting Parameters 69
Summary 70

PART 2: TAKING ADVANTAGE OF ILE 73

Chapter 6: BIF Up Your Code! 75
Get Rid of Those Annoying File-Related Indicators 75
Let's Start at the End—End of File, Actually 76
Still CHAINed to Indicators? 77
When %Equal Means More Efficient Code 78
Start Moving MOVE and MOVEL Out of Your Code 79
Removing MOVE from Your Numeric Conversion Operations 80
Need to Convert Something to Integer? Here's a (H)%int 81
Whatever %FLOATs Your Boat 82
A Word About Fixed-Format Code 84
Keep Moving MOVE Out of Your Code 84
Converting Numeric Data to Character with %CHAR 84
Using Edit Codes and Edit Words to "Beautify" Numeric Data 85
Yet Another Use of MOVE You Can Do Without 87
Simplifying String Operations with BIFs 88
"Ancient" Versus "Modern" Ways 89
Out with the Old, In with the New 90
Do You %TRIM? 91
Easily Find and Replace Text in Strings with BIFs 92
Making the "Modern" Approach More Dynamic with %SCAN 93
Improving It with %REPLACE 94
The Good and Bad News About %SCANRPL 96
Time to Build a Few BIFed Up Functions! 98
My Method for Building Procedures and Functions 98
Demonstrating the Methodology with a Few Functions 99
Building Excel-like String-Handling Functions 106
The LEFT Function 107

RIGHT Is Next 109
Building the Chg_Case Function 111
The Tricky Case of Proper Case 115
Using BIFs to Perform Date Operations 118
A Simple Problem: Calculating the Last Day of the Month 119
Introducing the %DATE BIF 119
The LastDayOfMonth Function 121
The %DIFF BIF: A Swiss Army Knife of Date Calculations 123
The Clc_DayOfWeek Function 124
The Rtv_DayOfWeek Function 126
Testing the Date-Related Functions 128
Using BIFs to Perform Time Operations 129
Do You Know How Long You Have to Work Until the Weekend? 130
This Chapter Doesn't Have a Summary 135

Chapter 7: Code Organization Strategies 137
When the Name Says It All: the Importance of Naming Conventions 137
Naming Variables, Part 1: Prefixes 138
Naming Variables, Part 2: Proper Names 143
Naming Procedures and Functions 144
Naming Modules 145
Naming Conventions for Physical and Logical Files 146
Using Prefixes as a Workaround for Duplicated Field Names in Multiple Files 147
Commenting and Documenting Strategies for Better Code 149
Documenting Procedures and Functions 149
Comment First, Code Later! 150
Documenting for the Lazy, I Mean Busy, Programmer 151
Defining Your Documentation Strategy 152
Code Organization 152
How to Organize Your Service Programs 153
Binding Directory Organization 154
Summary 156

Chapter 8: /FREE Your Code! 157
Free-Format Pros and Cons 157

It's /FREE, But It Has Rules 159
What the Most-Used Operation Codes Look Like in Free Format 160
Converting Fixed-Format Code to Free-Format 166
The Operation Codes Free Format Left Behind 167
The Quick Wins 167
Moving Toward a More Structured Programming Syntax 168
Operation Codes That Became BIFs 169
Other Operation Codes (Kind of) Replaced by BIFs 169
*Replacing COMP and CAS*XX 170
*Goodbye Spaghetti Code: No More CAB*XX*, TAG, and GOTOs* 171
Array-Related Operation Codes 172
Summary 176

Chapter 9: No More ISDB Nightmares: Meet the New ILE Debugger 181
The Interactive Source Debugger Versus the ILE Debugger 181
Getting to Know the Different Debugging Views 182
Choosing the Right View for You 188
Encrypting Your Debugging Views 189
Starting a Debug Session 190
Navigating in the Debug Session 193
The Actual Debug Process: Working with Breakpoints 196
The Actual Debug Process: Working with Watch Conditions 198
A Step-by-Step Debug Session 201
Debugging Service Programs 206
Using the ILE Debugger on OPM RPG and CL Programs 207
Debugging Batch Jobs 207
Summary 209

Chapter 10: The Latest and Greatest News for RPG 211
Free Your Code, from the First Line: the New H-specs 211
Free-Format File Definitions 213
Finally, Data Definitions in RPG Look "Modern"! 216
Make Your Variable Definitions Crystal-Clear 216
A New Way to Define Constants—and New Uses for Them 220
Simplifying the Data Structure Definition 222

What You Need to Know About the New Procedure Definitions 222
One Additional Example: Converting a Function Header to Full Free Format 224
Fully Free-Format Code 226
Summary 227

Chapter 11: SQL in a Nutshell **229**
What Is SQL? 229
A Concise Introduction to Data Manipulation Language 232
The Simplicity and Flexibility of the SELECT Statement 232
Adding New Rows with INSERT 240
"Massaging" Data with UPDATE 242
Try Not to Do (Too Much) Damage with DELETE 244
SQL's Column Functions: Adding Flexibility and Awesome Power to Your SQL Statements 245
Aggregate Functions 245
Scalar Functions 248
Tools at Your Disposal to Execute SQL Statements 270
Embedding SQL in Your RPG Code 279
How to Embed SQL Code in Fixed-Format RPG 280
How to Embed SQL Code in Free-Format RPG 281
How to Get RPG and SQL to Talk to One Another in Your Programs 281
How to Compile SQL-Infused RPG Code 285
Your First Embedded SQL Function 286
Using SQL Cursors to Replace Record-Level Data Access 289
Replacing the Dreaded OPNQRYF with an SQL Cursor 292
Ultra-Flexible Cursors: the Beauty of PREPARE 294
Other Embedded SQL Statements 296
Flipping It: Using RPG Code in SQL 299
Using ILE RPG Programs and Procedures as SQL SPs 300
Using ILE RPG Functions as SQL UDFs 303
A DDL Hands-on Tour 307
Providing a Parent for Your SQL Objects: Creating a Schema 308
Tables: Luxury Yachts for Your Data 310
Taking in the View 318

Using an Index to Improve Database Performance....320
OVRDBF Made Simple, Practical, and Permanent: the ALIAS SQL Instruction....322
SQL's Way to Delete Things: Drop Them....322
Simplifying Application Development with SQL Triggers....323
Summary....330

PART 3: BEYOND ILE— START MODERNIZING YOUR APPLICATIONS....335

Chapter 12: Modernizing Your Applications: Why, What, Where, and How....337
A Fuzzy Buzzword....337
Why You Should Give Modernization a Shot....338
Modernization Eye Candy....338
Restructuring the Database....339
The Big Question You Should Ask Yourself Before Starting a Modernization Process....341
The Benefits of Modernization for You, Your Boss, and Your Company....343
Tips to Avoid the Pains of Modernization....345
Setting Your Modernization Goals....347
Summary....354

Chapter 13: Database Modernization....357
A Bit of Database Theory....357
Conceptual, Logical, and Physical Models....358
Database Normalization....358
Entity Relationship Diagram....359
Tools to Help the Modernization Process....363
IBM Data Studio....363
IBM InfoSphere Data Architect....364
Adsero Optima Foundation....365
Database Modernization Methodology....365
Step One: Convert DDS Files to DDL Objects....366
Discover and List Existing Files....367

Figure Out and Map Implicit Relationships Between Files 367
Start Preparing to Tidy Up Your Database 368
Converting DDS to DDL 369
Normalizing the Database 371
Step Two: Move Business Rules to the Database 372
Putting Database Validation in the Database 372
Step Three: Take Advantage of DB2's Advanced Functionalities 386
Summary 391

Chapter 14: UI Modernization and the MVC Concept 393
Why Separate the UI Code from the Rest? 393
Building Your Programs the Modern Way 394
A Simple Multi-Tier Architecture Implementation 396
A More Realistic Multi-Tier Architecture Implementation 398
Good, But Not Enough—Introducing MVC 399
The Model Layer 401
The Controller Layer 403
The View Layer 404
Reengineering an ILE Program Using the MVC Design Pattern 404
RPG Open Access: UI Modernization Made Easy 405
Tools to Modernize the UI Using RPG OA 406
Summary 407
Where to Go from Here 408

Index 409