

Contents

Preface	v
1 Motivation	1
1.1 Why We Wrote This Book	1
1.2 For Whom We Are Writing	2
1.2.1 Those Accustomed to FORTRAN 77	2
1.2.2 Those Who Want to Create Windows Applications .	2
1.2.3 Those Who Want to Interface Fortran with Other Applications	3
1.3 The Need for Good Programming Practice	4
1.3.1 Programming for Research Dissemination	4
1.3.2 Programming Standards	4
1.3.3 Benefits of Good Programming Style	5
1.3.4 Benefits of Uniformity	5
1.4 Why We Use Fortran	6
1.4.1 History of Fortran	6
1.4.2 Fortran's Advantages	6
1.4.3 Useful New Features	7
1.4.4 What this book does not cover	7
1.4.5 Differences Between Fortran 90 and Fortran 95	7
1.4.6 Pseudo Object-Oriented Programming in Fortran . .	8
1.4.7 Fortran 2003	8
1.4.8 Which Compiler Should I Use?	8
1.5 Developing Applications for a Broad Audience	9

1.5.1	Console Applications and COM Servers	9
1.5.2	COM Servers and Clients	10
1.6	Scope of the Rest of This Book	11
1.7	Our Source Code and Web Site	12
2	Introduction to Modern Fortran	13
2.1	Getting Started	13
2.1.1	A Very Simple Program	13
2.1.2	Fixed and Free-Form Source Code	15
2.1.3	Compiling, Linking and Running	15
2.1.4	Compiler Options	17
2.1.5	Standard Input and Output	19
2.1.6	Intrinsic Uniform Generator	20
2.1.7	Integer and Real Kinds	20
2.1.8	Do, if, case, goto	22
2.1.9	Exercises	25
2.2	Arrays	26
2.2.1	Rank, Size and Shape	26
2.2.2	Array Functions	27
2.2.3	Operations on Arrays and Array Sections	28
2.2.4	Your Mileage May Vary	29
2.2.5	Array Allocation	30
2.2.6	Exercises	31
2.3	Basic Procedures	32
2.3.1	Subroutines	32
2.3.2	Assumed-Shape and Optional Arguments	33
2.3.3	Functions	36
2.3.4	Pure, Elemental and Recursive Procedures	38
2.3.5	On the Behavior of Local Variables	39
2.3.6	Exercises	41
2.4	Manipulating Character Strings	42
2.4.1	Character Variables	42
2.4.2	Assigning, Comparing, and Concatenating Strings	43
2.4.3	More String Functions	44
2.4.4	Internal Files	45
2.4.5	Exercises	46
2.5	Additional Topics	47
2.5.1	Expressions with Mixed Types and Kinds	47
2.5.2	Explicit Type Conversion	48
2.5.3	Generic Procedures	49
2.5.4	Don't Pause or Stop	50
2.6	Additional Exercises	50
3	A Pseudo Object-Oriented Style	55
3.1	Basic Concepts of Object-Oriented Programming	56

3.1.1	Objects and Classes	56
3.1.2	Properties	56
3.1.3	Put and Get	57
3.1.4	Methods and Constructors	57
3.1.5	Conceptualizing an Interface	58
3.1.6	Other Object-Oriented Concepts	60
3.1.7	Exercises	60
3.2	Modules	61
3.2.1	What Is a Module?	61
3.2.2	How Not to Use Modules	62
3.2.3	How to Use Modules	64
3.2.4	Generic Module Procedures	65
3.2.5	Exercises	66
3.3	Derived Types	67
3.3.1	What Is a Derived Type?	67
3.3.2	Using Derived Types	69
3.3.3	Constructors and Default Initialization	71
3.3.4	Exercises	73
3.4	Pointers	73
3.4.1	Fear Not the Pointer	73
3.4.2	Pointer Assignment	74
3.4.3	Pointer Status	76
3.4.4	Pointer Allocation	77
3.4.5	Pointer Deallocation	77
3.4.6	Memory Leaks	78
3.4.7	Exercises	79
3.5	Why We Need Pointers	79
3.5.1	Pointers in Derived Types	79
3.5.2	Pointers as Dummy Arguments	80
3.5.3	Recursive Data Structures	82
3.5.4	Procedures for Linked Lists	83
3.5.5	Exercises	86
3.6	Example Module: A Generic Error Handler	88
3.6.1	Strategy for Managing Run-Time Errors	88
3.6.2	Structure of the Module	89
3.6.3	Module Procedures	91
3.6.4	Using the Module	96
3.6.5	General Guidelines for Modules	99
3.7	Additional Exercises	100
4	Implementing Computational Routines	101
4.1	Issues of Numerical Accuracy	101
4.1.1	Accuracy Is Crucial	101
4.1.2	Floating-Point Approximation	102
4.1.3	Roundoff and Cancellation Error	103

4.1.4	Arithmetic Exceptions	104
4.1.5	Resources for Numerical Programming	106
4.1.6	Exercises	106
4.2	Example: Fitting a Simple Finite Mixture	109
4.2.1	The Problem	109
4.2.2	Programming Constants	110
4.2.3	A Computational Engine Module	111
4.2.4	A Public Procedure With Safeguards	112
4.2.5	The Computations	114
4.2.6	Strategies for Calling the Engine	115
4.2.7	A Simple Calling Program	116
4.2.8	Test, and Test Again	118
4.2.9	Exercises	119
4.3	Efficient Routines at Lower Levels	120
4.3.1	What Is a Lower-Level Procedure?	120
4.3.2	Keeping Overhead Low	121
4.3.3	Taking Advantage of Structure	122
4.3.4	Loop Reordering, Stride and Saxpy	123
4.3.5	Optimization, Pipelining and Multiple Processors	125
4.3.6	A Simple Example	126
4.3.7	Hidden Allocation of Temporary Arrays	127
4.3.8	Exercises	128
4.4	More Computational Procedure Examples	128
4.4.1	An Improved Cholesky Factorization	128
4.4.2	Inverting a Symmetric Positive-Definite Matrix	130
4.4.3	Weighted Least Squares	132
4.4.4	Computational Routines in Object-Oriented Programming	134
4.5	Additional Exercises	136
5	Developing a Console Application	139
5.1	A Program for Logistic Regression	140
5.1.1	The logistic regression model	140
5.1.2	Motivation for the ELOGIT Console Program	140
5.1.3	Dependency Map for Source Components	140
5.1.4	Developing Program Units Incrementally	141
5.2	Where to Begin	142
5.2.1	Before You Start	142
5.2.2	Program Constants	143
5.2.3	The Control File Interface	144
5.2.4	First Snapshot of ELOGIT	148
5.2.5	Exercises	150
5.3	Starting the Main Types Module	151
5.3.1	An Object-Oriented Design	151
5.3.2	Storing the Dataset	152

5.3.3	A Module for Pointer Allocation	156
5.3.4	Putting and Getting Data	158
5.3.5	Reading Data from Files	163
5.3.6	Second Snapshot of ELOGIT	165
5.3.7	Exercises	167
5.4	Specifying the Model	167
5.4.1	Storing the Model Specification	167
5.4.2	Putting and Getting Model Properties	168
5.4.3	Third Snapshot	172
5.4.4	Exercises	175
5.5	Fitting the Model	176
5.5.1	The Computational Task	176
5.5.2	Newton-Raphson and Weighted Least Squares	176
5.5.3	Parameters and Results	178
5.5.4	The Model-Fitting Procedure	179
5.5.5	Reporting the Results	183
5.5.6	Looking Ahead	188
5.6	Additional Exercises	188
6	Creating and Using Dynamic-Link Libraries	191
6.1	Extending the Functionality of Statistical Packages with Fortran DLLs	191
6.1.1	Compiled Procedures Run Faster	191
6.1.2	When to Use a DLL	192
6.2	Understanding Libraries	193
6.2.1	Source-Code Libraries	193
6.2.2	Static Libraries	194
6.2.3	Dynamic-Link Libraries	194
6.3	How Programs Use DLLs	195
6.3.1	Locating the DLL	195
6.3.2	DLL Hell	196
6.3.3	Dynamic Loading and Linking	196
6.3.4	Load-Time and Run-Time Linking	197
6.4	Creating a Fortran DLL	198
6.4.1	The Basic Steps	198
6.4.2	Passing Arguments	198
6.4.3	Calling Conventions	199
6.4.4	Compiling and Linking the Source Code	201
6.4.5	Compiler Options	202
6.5	Example: a Fortran DLL for Fitting an Exponential Mixture	202
6.5.1	Creating a Wrapper	202
6.5.2	Building the DLL with Intel Visual Fortran and Lahey/Fujitsu Fortran	207
6.5.3	Building with Salford Fortran	209
6.5.4	Calling the DLL Procedure from S-PLUS and R	211

6.5.5	Calling the Function from SAS/IML	214
6.6	Shared Objects in Unix and Linux	216
6.6.1	An Example: Extending S-Plus and R via a Fortran Shared Object in Linux	217
7	Creating COM Servers	219
7.1	A Simple Example	220
7.1.1	The magic8 Fortran Module	220
7.1.2	The Magic8 COM Server	222
7.1.3	Installing the Magic8 COM Server	222
7.1.4	The 8-Ball Speaks in Excel	223
7.1.5	The 8-Ball Speaks in S-PLUS and R	225
7.1.6	The 8-Ball Speaks in MATLAB	226
7.1.7	The 8-Ball Speaks in SAS	227
7.1.8	Exercises	228
7.2	COM Server Basics	228
7.2.1	References on COM	228
7.2.2	COM, Windows, and .NET	228
7.2.3	COM Servers versus Conventional DLLs	229
7.2.4	The Object-Oriented Contract	230
7.2.5	In-Process versus Out-of-Process Servers	231
7.3	Example: A COM Server for Logistic Regression	233
7.3.1	Producing COM Servers with Intel Visual Fortran	233
7.3.2	Getting Ready	233
7.3.3	Naming the Server and the Class	234
7.3.4	Fortran Style Conventions	235
7.3.5	Automatically Generating the COM Server Code	236
7.3.6	Building the Project in Visual Studio	237
7.3.7	Building and registering the server	243
7.3.8	Creating a Simple Client	245
7.4	Exercises	248
7.5	How the Fortran COM Server Works	249
7.5.1	Overview of the Automatically Generated Code	249
7.5.2	The Interface Definition Language File	250
7.5.3	The Instance Code	250
7.5.4	The Interface Code	253
7.5.5	Passing Arrays as Variants	255
7.5.6	How the COM Server Handles Errors	259
7.6	Distributing and Installing COM Servers	260
7.7	Additional Exercises	262
8	Creating COM Clients	265
8.1	An Improved Client for Excel	265
8.1.1	Excel As a Graphical User Interface	265
8.1.2	Starting to Write the Client	266

- 8.1.3 How Did It Work? 269
- 8.1.4 Debugging the Client and Server 270
- 8.1.5 Finishing the Excel Client 274
- 8.1.6 Exercises 277
- 8.2 Clients for Other Environments 277
 - 8.2.1 Keeping It Simple 277
 - 8.2.2 Clients for S-PLUS and R 277
 - 8.2.3 A Client for SAS 281
 - 8.2.4 SPSS 288
 - 8.2.5 MATLAB 293
- 8.3 Creating a Standalone GUI Application 296
 - 8.3.1 Component-Based Design 296
 - 8.3.2 Visual Basic .NET 296
 - 8.3.3 An XML Data Format 296
 - 8.3.4 Starting the Graphical Interface 298
 - 8.3.5 Reading the Data File 299
 - 8.3.6 Specifying the Model 301
 - 8.3.7 Invoking the Model Fit Procedure 304
 - 8.3.8 Displaying the Results 308
 - 8.3.9 Finishing Up 312
 - 8.3.10 Exercises 312

References 315