

contents

<i>foreword</i>	<i>xi</i>
<i>preface</i>	<i>xiii</i>
<i>acknowledgments</i>	<i>xiv</i>
<i>about this book</i>	<i>xvi</i>
<i>about the author</i>	<i>xxii</i>
<i>about the cover illustration</i>	<i>xxiii</i>

PART 1 GETTING STARTED WITH MODERN FORTRAN.....1

<i>I</i>	<i>Introducing Fortran</i>	<i>3</i>
1.1	What is Fortran?	4
1.2	Fortran features	6
1.3	Why learn Fortran?	8
1.4	Advantages and disadvantages	10
	<i>Side-by-side comparison with Python</i>	<i>10</i>
1.5	Parallel Fortran, illustrated	12
1.6	What will you learn in this book?	13
1.7	Think parallel!	14
	<i>Copying an array from one processor to another</i>	<i>17</i>

- 1.8 Running example: A parallel tsunami simulator 22
 - Why tsunami simulator?* 22
 - *Shallow water equations* 23
 - What we want our app to do* 24
- 1.9 Further reading 25

2 **Getting started: Minimal working app** 26

- 2.1 Compiling and running your first program 27
- 2.2 Simulating the motion of an object 28
 - What should our app do?* 29
 - *What is advection?* 30
- 2.3 Implementing the minimal working app 31
 - Implementation strategy* 32
 - *Defining the main program* 33
 - Declaring and initializing variables* 34
 - *Numeric data types* 35
 - Declaring the data to use in our app* 37
 - *Branching with an if block* 40
 - *Using a do loop to iterate* 42
 - *Setting the initial water height values* 44
 - *Predicting the movement of the object* 45
 - *Printing results to the screen* 47
 - *Putting it all together* 47
- 2.4 Going forward with the tsunami simulator 51
- 2.5 Answer key 52
 - Exercise: Cold front propagation* 52
- 2.6 New Fortran elements, at a glance 52
- 2.7 Further reading 52

PART 2 CORE ELEMENTS OF FORTRAN55

3 **Writing reusable code with functions and subroutines** 57

- 3.1 Toward higher app complexity 58
 - Refactoring the tsunami simulator* 58
 - *Revisiting the cold front problem* 61
 - *An overview of Fortran program units* 63
- 3.2 Don't repeat yourself, use procedures 65
 - Your first function* 65
 - *Expressing finite difference as a function in the tsunami simulator* 70
- 3.3 Modifying program state with subroutines 72
 - Defining and calling a subroutine* 72
 - *When do you use a subroutine over a function?* 74
 - *Initializing water height in the tsunami simulator* 75

- 3.4 Writing pure procedures to avoid side effects 76
 - What is a pure procedure?* 76
 - *Some restrictions on pure procedures* 77
 - *Why are pure functions important?* 77
- 3.5 Writing procedures that operate on both scalars and arrays 77
- 3.6 Procedures with optional arguments 79
- 3.7 Tsunami simulator: Putting it all together 81
- 3.8 Answer key 82
 - Exercise 1: Modifying state with a subroutine* 82
 - *Exercise 2: Writing an elemental function that operates on both scalars and arrays* 83
- 3.9 New Fortran elements, at a glance 83
- 3.10 Further reading 84

4 Organizing your Fortran code using modules 85

- 4.1 Accessing a module 86
 - Getting compiler version and options* 86
 - *Using portable data types* 89
- 4.2 Creating your first module 91
 - The structure of a custom module* 92
 - *Defining a module* 93
 - Compiling Fortran modules* 95
 - *Controlling access to variables and procedures* 97
 - *Putting it all together in the tsunami simulator* 98
- 4.3 Toward realistic wave simulations 99
 - A brief look at the physics* 101
 - *Updating the finite difference calculation* 102
 - *Renaming imported entities to avoid name conflict* 104
 - *The complete code* 105
- 4.4 Answer key 107
 - Exercise 1: Using portable type kinds in the tsunami simulator* 107
 - Exercise 2: Defining the set_gaussian subroutine in a module* 107
- 4.5 New Fortran elements, at a glance 108
- 4.6 Further reading 108

5 Analyzing time series data with arrays 110

- 5.1 Analyzing stock prices with Fortran arrays 111
 - Objectives for this exercise* 111
 - *About the data* 112
 - Getting the data and code* 114

- 5.2 Finding the best and worst performing stocks 114
Declaring arrays 116 ▪ *Array constructors* 118 ▪ *Reading stock data from files* 121 ▪ *Allocating arrays of a certain size or range* 122 ▪ *Allocating an array from another array* 123
Automatic allocation on assignment 123 ▪ *Cleaning up after use* 124 ▪ *Checking for allocation status* 126 ▪ *Catching allocation and deallocation errors* 126 ▪ *Implementing the CSV reader subroutine* 127 ▪ *Indexing and slicing arrays* 129
- 5.3 Identifying risky stocks 132
- 5.4 Finding good times to buy and sell 135
- 5.5 Answer key 139
Exercise 1: Convenience (de)allocator subroutines 139 ▪ *Exercise 2: Reversing an array* 140 ▪ *Exercise 3: Calculating moving average and standard deviation* 140
- 5.6 New Fortran elements, at a glance 141
- 5.7 Further reading 141
- 6 Reading, writing, and formatting your data 143**
- 6.1 Your first I/O: Input from the keyboard and output to the screen 144
The simplest I/O 144 ▪ *Reading and writing multiple variables at once* 147 ▪ *Standard input, output, and error* 148
- 6.2 Formatting numbers and text 151
Designing the aircraft dashboard 151 ▪ *Formatting strings, broken down* 152 ▪ *Format statements in legacy Fortran code* 157
- 6.3 Writing to files on disk: A minimal note-taking app 157
Opening a file and writing to it 158 ▪ *Opening a file* 159
Writing to a file 161 ▪ *Appending to a file* 162 ▪ *Opening files in read-only or write-only mode* 163 ▪ *Checking whether a file exists* 164 ▪ *Error handling and closing the file* 167
- 6.4 Answer key 168
Exercise: Redirect stdout and stderr to files 168
- 6.5 New Fortran elements, at a glance 169

PART 3 ADVANCED FORTRAN USE 171

7 Going parallel with Fortran coarrays 173

- 7.1 Why write parallel programs? 174
- 7.2 Processing real-world weather buoy data 175
 - About the data* 176 ▪ *Getting the data and code* 178
 - Objectives* 178 ▪ *Serial implementation of the program* 179
- 7.3 Parallel processing with images and coarrays 181
 - Fortran images* 182 ▪ *Getting information about the images* 183
 - Telling images what to do* 184 ▪ *Gathering all data to a single image* 186
- 7.4 Coarrays and synchronization, explained 187
 - Declaring coarrays* 188 ▪ *Allocating dynamic coarrays* 188
 - Sending and receiving data* 189 ▪ *Controlling the order of image execution* 191
- 7.5 Toward the parallel tsunami simulator 192
 - Implementation strategy* 192 ▪ *Finding the indices of neighbor images* 194
 - *Allocating the coarrays* 195 ▪ *The main time loop* 196
- 7.6 Answer key 199
 - Exercise 1: Finding the array subranges on each image* 199
 - Exercise 2: Writing a function that returns the indices of neighbor images* 200
- 7.7 New Fortran elements, at a glance 201
- 7.8 Further reading 201

8 Working with abstract data using derived types 202

- 8.1 Recasting the tsunami simulator with derived types 203
- 8.2 Defining, declaring, and initializing derived types 206
 - Defining a derived type* 209 ▪ *Instantiating a derived type* 210
 - Accessing derived type components* 212 ▪ *Positional vs. keyword arguments in derived type constructors* 212
 - *Providing default values for derived type components* 214 ▪ *Writing a custom type constructor* 215
 - *Custom type constructor for the Field type* 218
- 8.3 Binding procedures to a derived type 220
 - Your first type-bound method* 220 ▪ *Type-bound methods for the Field type* 221
 - *Controlling access to type components and methods* 222 ▪ *Bringing it all together* 224

- 8.4 Extending tsunami to two dimensions 224
 - Going from 1-D to 2-D arrays* 225 ▪ *Updating the equation set* 226 ▪ *Finite differences in x and y* 226 ▪ *Passing a class instance to diffx and diffy functions* 228 ▪ *Derived type implementation of the tsunami solver* 229
- 8.5 Answer key 231
 - Exercise 1: Working with private components* 231 ▪ *Exercise 2: Invoking a type-bound method from an array of instances* 233
 - Exercise 3: Computing finite difference in y direction.* 233
- 8.6 New Fortran elements, at a glance 234
- 8.7 Further reading 235

9 **Generic procedures and operators for any data type** 236

- 9.1 Analyzing weather data of different types 237
 - About the data* 238 ▪ *Objectives* 241 ▪ *Strategy for this exercise* 242
- 9.2 Type systems and generic procedures 242
 - Static versus strong typing* 242
- 9.3 Writing your first generic procedure 243
 - The problem with strong typing* 243 ▪ *Writing the specific functions* 244 ▪ *Writing the generic interface* 247 ▪ *Results and complete program* 251
- 9.4 Built-in and custom operators 253
 - What's an operator?* 253 ▪ *Things to do with operators* 253
 - Fortran's built-in operators* 255 ▪ *Operator precedence* 257
 - Writing custom operators* 257 ▪ *Redefining built-in operators* 258
- 9.5 Generic procedures and operators in the tsunami simulator 259
 - Writing user-defined operators for the Field type* 259
- 9.6 Answer key 260
 - Exercise 1: Specific average function for a derived type* 260
 - Exercise 2: Defining a new string concatenation operator* 262
- 9.7 New Fortran elements, at a glance 263

10 **User-defined operators for derived types** 264

- 10.1 Happy Birthday! A countdown app 265
 - Some basic specification* 265 ▪ *Implementation strategy* 266

- 10.2 Getting user input and current time 266
 - Your first datetime class* 266
 - Reading user input* 267
 - Getting current date and time* 271
- 10.3 Calculating the difference between two times 272
 - Modeling a time interval* 273
 - Implementing a custom subtraction operator* 273
 - Time difference algorithm* 275
 - The complete program* 280
- 10.4 Overriding operators in the tsunami simulator 282
 - A refresher on the Field class* 283
 - Implementing the arithmetic for the Field class* 284
 - Synchronizing parallel images on assignment* 286
- 10.5 Answer key 288
 - Exercise 1: Validating user input* 288
 - Exercise 2: Leap year in the Gregorian calendar* 289
 - Exercise 3: Implementing the addition for the Field type* 289
- 10.6 New Fortran elements, at a glance 290

PART 4 THE FINAL STRETCH.....291

11 *Interoperability with C: Exposing your app to the web* 293

- 11.1 Interfacing C: Writing a minimal TCP client and server 294
 - Introducing networking to Fortran* 295
 - Installing libdill* 297
- 11.2 TCP server program: Receiving network connections 297
 - IP address data structures* 299
 - Initializing the IP address structure* 301
 - Checking IP address values* 306
 - Intermezzo: Matching compatible C and Fortran data types* 308
 - Creating a socket and listening for connections* 310
 - Accepting incoming connections to a socket* 311
 - Sending a TCP message to the client* 312
 - Closing a connection* 315
- 11.3 TCP client program: Connecting to a remote server 317
 - Connecting to a remote socket* 317
 - Receiving a message* 319
 - The complete client program* 321
- 11.4 Some interesting mixed Fortran-C projects 322
- 11.5 Answer key 322
 - Exercise 1: The Fortran interface to ipaddr_port* 322
 - Exercise 2: Fortran interfaces to suffix_detach and tcp_close* 323

- 11.6 New Fortran elements, at a glance 324
- 11.7 Further reading 324

12 *Advanced parallelism with teams, events, and collectives* 326

- 12.1 From coarrays to teams, events, and collectives 327
- 12.2 Grouping images into teams with common tasks 328
 - Teams in the tsunami simulator* 329
 - *Forming new teams* 331
 - Changing execution between teams* 332
 - *Synchronizing teams and exchanging data* 335
- 12.3 Posting and waiting for events 338
 - A push notification example* 339
 - *Posting an event* 341
 - Waiting for an event* 341
 - *Counting event posts* 342
- 12.4 Distributed computing using collectives 343
 - Computing the minimum and maximum of distributed arrays* 343
 - Collective subroutines syntax* 345
 - *Broadcasting values to other images* 346
- 12.5 Answer key 347
 - Exercise 1: Hunters and gatherers* 347
 - *Exercise 2: Tsunami time step logging using events* 350
 - *Exercise 3: Calculating the global mean of water height* 351
- 12.6 New Fortran elements, at a glance 353
- 12.7 Further reading 353

appendix A *Setting up the Fortran development environment* 355

appendix B *From calculus to code* 361

appendix C *Concluding remarks* 366

index 381