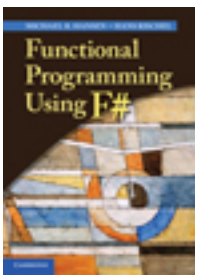

Functional Programming Using F#

Michael R. Hansen, Hans Rischel

Book DOI: http://dx.doi.org/10.1017/CBO9781139093996

Online ISBN: 9781139093996

Hardback ISBN: 9781107019027

Paperback ISBN: 9781107684065

Chapter

Preface pp. ix-xii

Chapter DOI: http://dx.doi.org/10.1017/CBO9781139093996.001

Cambridge University Press

Preface

The purpose of this book is to introduce a wide range of readers – from the professional programmer to the computer science student – to the rich world of functional programming using the F# programming language. The book is intended as the textbook in a course on functional programming and aims at showing the role of functional programming in a wide spectrum of applications ranging from computer science examples over database examples to systems that engage in a dialogue with a user.

Why functional programming using F#?

Functional programming languages have existed in academia for more than a quarter of a century, starting with the untyped Lisp language, followed by strongly typed languages like Haskell and Standard ML.

The penetration of functional languages to the software industry has, nevertheless, been surprisingly slow. The reason is probably lack of support of functional languages by commercial software development platforms, and software development managers are reluctant to base software development on languages living in a non-commercial environment.

This state of affairs has been changed completely by the appearance of F#, an open-source, full-blown functional language integrated in the Visual Studio development platform and with access to all features in the .NET program library. The language is also supported on Linux and MAC systems using the Mono platform.

The background

The material in this book has been developed in connection with courses taught at the Technical University of Denmark, originating from the textbook *Introduction to Programming Using SML* by Hansen and Rischel (Addison-Wesley, 1999).

It has been an exciting experience for us to learn the many elegant and useful features of the F# language, and this excitement is hopefully transferred to the reader of this book.

The chapters

- Chapter 1: The basic concepts of F#, including values, types and recursive functions, are introduced in a manner that allows readers to solve interesting problems from the start.
- Chapter 2: A thorough introduction to the basic types in F# is given, together with a gentle introduction to the notion of higher-order functions.
- Chapter 3: The simplest composite types of F#, tuples and records, are introduced. They allow several values to be grouped together into one component. Furthermore, tagged values are introduced.

- Chapter 4: A list is a finite sequence of values with the same type. Standard recursions on lists are studied and examples illustrating a model-based approach to functional programming are given.
- Chapter 5: The concepts of sets and maps are introduced and the powerful F# collection libraries for lists, sets and maps are studied and applied in connection with a model-based approach.
- Chapter 6: The concept of finite tree is introduced and illustrated through a broad selection of examples.
- Chapter 7: It is shown how users can make their own libraries by means of modules consisting of signature and implementation files. Furthermore, object-oriented features of F# are mentioned.
- Chapter 8: Imperative features of F# are introduced, including the array part of the collection library and the imperative sets and maps from the .NET framework.
- Chapter 9: The memory management concepts, stack, heap and garbage collection, are described. Tail-recursive functions are introduced and two techniques for deriving such functions are presented: one using accumulating parameters, the other continuations. Their efficiency advantages are illustrated.
- Chapter 10: A variety of facilities for processing text are introduced, including regular expressions, file operations, web-based operations and culture-dependent string ordering. The facilities are illustrated using a real-world example.
- Chapter 11: A sequence is a, possibly infinite, collection of elements that are computed on-demand only. Sequence functions are expressed using library functions or sequence expressions that provide a step-by-step method for generating elements. Database tables are viewed as sequences (using a type provider) and operations on databases are expressed using query expressions.
- Chapter 12: The notion of computation expression, which is based on the theory of monads, is studied and used to hide low-level details of a computation from its definition. Monadic parsing is used as a major example to illustrate the techniques.
- Chapter 13: This last chapter describes how to construct asynchronous reactive programs, spending most of their time awaiting a request or a response from an external agent, and parallel programs, exploiting the multi-core processor of the computer.

The first six chapters cover a standard curriculum in functional programming, while the other chapters cover more advanced topics.

Further material

The book contains a large number of exercises, and further material is available at the book's homepage. A link to this homepage is found at:

`http://www.cambridge.org/9781107019027`

This material includes a complete set of slides for a course in functional programming plus a collection of problems and descriptions of topics to be used in student projects.

Acknowledgments

Special thanks go to Peter Sestoft, Don Syme and Anh-Dung Phan. The idea to make a textbook on functional programming on the basis of F# originates from Peter, who patiently commented on the manuscript during its production and helped with advice and suggestions. From the very start of this project we had the support of Don. This is strongly appreciated and so is the help, clarifications and constructive comments that we received throughout this project. Phan helped with many comments, suggestions and insights about the platform. We are grateful for this help, for many discussions and for careful comments on all the chapters.

Furthermore, we are grateful to Nils Andersen, Mary E. Böker, Diego Colombo and Niels Hallenberg for reading and commenting on the complete manuscript.

Earlier versions of this manuscript have been used in connection with courses at the Technical University of Denmark and the IT-University of Copenhagen. The comments we received from the students in these courses are greatly appreciated.

Lyngby, July 31, 2012
Michael R. Hansen and Hans Rischel