Functional Programming in Clean

Preface

Functional languages enable programmers to concentrate on the problem one would like to solve without being forced to worry about all kinds of uninteresting implementation details. A functional program can be regarded as an executable specification. Functional programming languages are therefore very popular in educational and research environments. Functional languages very suited for teaching students the first principles of programming. In research environments they are used for rapid prototyping of complex systems. Recent developments in the implementation techniques and new insights in the underlying concepts such as input/output handling make that modern functional languages can nowadays also be used successfully for the development of real world applications.

The purpose of this book is to teach practical programming skills using the state-of-the art pure functional language Concurrent Clean. Clean has many aspects in common with other modern functional languages like Miranda, Haskell and ML. In addition Clean offers additional support for the development of window based applications as well as support for process communication and for the development of (distributed) applications.

The book is split into three parts.

In the first part of this book an introduction into functional programming is given. In five Chapters we treat the basic aspects of functional programming, functions, data structures, the type system and I/O handling. The target here is that you are able to write functions and programs as soon as possible. A complete description of all allowed language constructs can be found in the Clean language manual.

The main emphasis of this book lies in the second part in which eleven case studies are presented. Each case treats a tiny, but complete application in an illustrative problem domain. Each case furthermore illustrates a certain aspect of functional programming. Some case applications are reused in others to illustrated the reusability of code.

The following cases are presented:

1. a simple database, illustrating simple file I/O and the construction of dynamically configurable dialogue interfaces;
2. a relational database (based on the simple database) illustrating the basic aspects of a relational database, showing the use of functional languages as executable specification as well as the reusability of code;
3. a window based text editor, illustrating the basic way to do window based text handling;

4. a window based graphical editor, illustrating the basic aspects of window based handling of graphical elements as well as given an example of an object oriented style of functional programming;
5. parser combinators, which will give a good insight in the different kind of parsing techniques, while we also illustrate a combinatorial style of functional programming;
6. an interpreter for a functional programming language for a subset of Clean (using the parser combinator tool of Chapter 5 and the text editor of Chapter 3), illustrating the basic semantics of functional languages and the reusability of code;
7. a spreadsheet (reusing the interpreter of Chapter 6) in which we also show the difference between interpreted and compiled code;
8. a computer architecture and programming language, in which we show how a functional language can be used as executable specification of an abstract machine architecture; also an assembler and imperative programming language is defined on top of the machine architecture showing the compilation path from high level language, through assembly language, into machine language.
9. compression and decompression of data, in which we show how the change an executable specification for a data compression / decompression algorithm into an efficient functional application.
10. process control, in which the processes and process communication is illustrated;
11. a distributed organizer, in which the use of distributed processes is shown.

In the third part of this book we discuss the different kinds of programming development techniques for functional programming (Chapter 1) and finally we treat efficiency aspects (Chapter 2).

So, a lot of material is presented in this book. However, one certainly does not have to work through all case studies. Depending on the programming experience already acquired and the time available one can use this book as a textbook for or one or two semester course. The book can be used as an introductory textbook for people with little programming experience. It can also be used for people who already have programming experience in other programming paradigm (imperative, object-oriented or logical) and now want to learn how to develop applications in a pure functional language.

We hope that you enjoy the book and that it will stimulate you to use a functional language for the development of your applications.

Table of Contents