1. Record Nr. UNINA9910784235103321

Autore	Dominus Mark Jason <1969->
Titolo	Higher-order Perl [[electronic resource]] : a guide to program transformation / / Mark Jason Dominus
Pubbl/distr/stampa	Amsterdam ; ; Boston, Mass., : Morgan Kaufmann Publishers, c2005
ISBN	1-281-01008-1 9786611010089 1-4237-0817-2 0-08-047834-4
Edizione	[1st ed.]
Descrizione fisica	1 online resource (601 p.)
Disciplina	005.13/3
Soggetti	Perl (Computer program language) Object-oriented programming (Computer science)
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Includes indexes.
Nota di contenuto	Front Cover; Higher-Order Perl: A Guide to Program Transformation; Copyright Page; Contents; Preface; Chapter 1. Recursion and Callbacks; 1.1 Decimal to Binary Conversion; 1.2 Factorial; 1.3 The Tower of Hanoi; 1.4 Hierarchical Data; 1.5 Applications and Variations of Directory Walking; 1.6 Functional Versus Object-Oriented Programming; 1.7 HTML; 1.8 When Recursion Blows Up; Chapter 2. Dispatch Tables; 2.1 Configuration File Handling; 2.2 Calculator; Chapter 3. Caching and Memoization; 3.1 Caching Fixes Recursion; 3.2 Inline Caching; 3.3 Good Ideas; 3.4 Memoization; 3.5 The Memoize Module 3.6 Caveats3.7 Key Generation; 3.8 Caching in Object Methods; 3.9 Persistent Caches; 3.10 Alternatives to Memoization; 3.11 Evangelism; 3.12 The Benefits of Speed; Chapter 4. Iterators; 4.1 Introduction; 4.2 Homemade Iterators; 4.3 Examples; 4.4 Filters and Transforms; 4.5 The Semipredicate Problem; 4.6 Alternative Interfaces to Iterators; 4.7 An Extended Example: Web Spiders; Chapter 5. From Recursion to Iterators; 5.1 The Partition Problem Revisited; 5.2 How to Convert a Recursive Function to an Iterator; 5.3 A Generic Search Iterator; 5.4 Other General Techniques for Eliminating Recursion

Chapter 6. Indinite Streams6.1 Linked Lists; 6.2 Lazy Linked Lists; 6.3 Recursive Streams; 6.4 The Hamming Problem; 6.5 Regex String Generation; 6.6 The Newton-Raphson Method; 6.7 Power Series; Chapter 7. Higher-Order Functions and Currying; 7.1 Currying; 7.2 Common Higher-Order Functions; 7.3 reduce() and combine(); 7.4 Databases; Chapter 8. Parsing; 8.1 Lexers; 8.2 Parsing in General; 8.3 Recursive-Descent Parsers; 8.4 Arithmetic Expressions; 8.5 Parsing Regexes; 8.6 Outlines; 8.7 Database-Query Parsing; 8.8 Backtracking Parsers; 8.9 Overloading; Chapter 9. Declarative Programming 9.1 Constraint Systems9.2 Local Propagation Networks; 9.3 Linear Equations; 9.4 linogram: A Drawing System; 9.5 Conclusion; Index; Function Index; A Note About the Cover

Sommario/riassunto

Most Perl programmers were originally trained as C and Unix programmers, so the Perl programs that they write bear a strong resemblance to C programs. However, Perl incorporates many features that have their roots in other languages such as Lisp. These advanced features are not well understood and are rarely used by most Perl programmers, but they are very powerful. They can automate tasks in everyday programming that are difficult to solve in any other way. One of the most powerful of these techniques is writing functions that manufacture or modify other functions. For example, instead of wri