1. Record Nr. UNINA9910784454503321

Autore	Robinson John A
Titolo	Software design for engineers and scientists [[electronic resource] /] / John A. Robinson
Pubbl/distr/stampa	Amsterdam ; ; Boston, : Newnes, 2004
ISBN	1-281-00327-1 9786611003272 0-08-047440-3
Edizione	[1st edition]
Descrizione fisica	1 online resource (429 p.)
Disciplina	005.1 005.102462
Soggetti	Computer software - Development Engineering - Data processing Computer programming Software engineering
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Description based upon print version of record.
Nota di bibliografia	Includes bibliographical references and index.
Nota di contenuto	Cover; Software Design for Engineers and Scientists; Contents; Preface; Acknowledgements; Errors; 1 Introduction; 1.1 Theme; 1.2 Audience; 1.3 Three definitions and a controversy; 1.4 Essential software design; 1.5 Outline of the book; Foundations; Software technology; Applied software design; Case studies; 1.6 Presentation conventions; 1.7 Chapter end material; Bibliography; 2 Fundamentals; 2.1 Introduction; 2.2 The nature of software; 2.3 Software as mathematics; 2.4 Software as literature; 2.5 Organic software; 2.6 Software design as engineering; 2.7 Putting the program in its place 2.8 User-centred design2.9 The craft of program construction; 2.10 Programmers' programming; 2.11 Living with ambiguity; 2.12 Summary; 2.13 Chapter end material; Bibliography; 3 The craft of software design; 3.1 Introduction; 3.2 Collaboration and imitation; 3.3 Finishing; 3.4 Tool building; 3.5 Logbooks; 3.6 The personal library; 3.7 Chapter end material; Bibliography; 4 Beginning programming in C++; 4.1 Introduction; 4.2 The programming environment; 4.3 Program shape, output, and the basic types; 4.4 Variables and their

types; 4.5 Conditionals and compound statements; 4.6 Loops
4.7 Random numbers, timing and an arithmetic game4.8 Functions; 4.9 Arrays and C-strings; 4.10 Program example: A dice-rolling simulation; 4.11 Bitwise operators; 4.12 Pointers; 4.13 Arrays of pointers and program arguments; 4.14 Static and global variables; 4.15 File input and output; 4.16 Structures; 4.17 Pointers to structures; 4.18 Making the program more general; 4.19 Loading structured data; 4.20 Memory allocation; 4.21 typedef; 4.22 enum; 4.23 Mechanisms that underlie the program; 4.24 More on the C/C++ standard library; 4.25 Chapter end material; Bibliography
5 Object-oriented programming in C++5.1 The motivation for object-oriented programming; Objects localize information; In an object-oriented language, existing solutions can be extended powerfully; 5.2 Glossary of terms in object-oriented programming; Data structure; Abstract Data Type (ADT); Class; Object; Method; Member function; Message; Base types and derived types; Inheritance; Polymorphism; 5.3 C++ type definition, instantiation and using objects; Stack ADT example; Location ADT example; Vector ADT example; 5.4 Overloading; Operator overloading; 5.5 Building a String class
5.6 Derived types, inheritance and polymorphismLocations and mountains example; Student marks example; 5.7 Exceptions; 5.8 Templates; 5.9 Streams; 5.10 C++ and information localization; 5.11 Chapter end material; Bibliography; 6 Program style and structure; 6.1 Write fewer bugs!; 6.2 Ten programming errors and how to avoid them; The invalid memory access error; The off-by-1 error; Incorrect initialization; Variable type errors; Loop errors; Incorrect code blocking; Returning a pointer or a reference to a local variable; Other problems with new and delete; Inadequate checking of input data
Different modules interpret shared items differently

Sommario/riassunto

Software Design for Engineers and Scientists integrates three core areas of computing:. Software engineering - including both traditional methods and the insights of 'extreme programming'. Program design - including the analysis of data structures and algorithms. Practical object-oriented programmingWithout assuming prior knowledge of any particular programming language, and avoiding the need for students to learn from separate, specialised Computer Science texts, John Robinson takes the reader from small-scale programing to competence in large software projects, all within