1.	Record Nr.	UNINA9910151658003321
	Autore	Deitel Harvey
	Titolo	C++ How to Program (Early Objects Version), International Edition: Early Objects Version
	Pubbl/distr/stampa	[Place of publication not identified], : Pearson Education Limited, 2013
	ISBN	0-273-79360-8
	Edizione	[9th ed.]
	Descrizione fisica	1 online resource (1064 pages)
	Lingua di pubblicazione	Inglese
	Formato	Materiale a stampa
	Livello bibliografico	Monografia
	Note generali	Bibliographic Level Mode of Issuance: Monograph
	Nota di contenuto	Cover -- Contents -- Preface -- 1 Introduction to Computers and C++ -- 1.1 Introduction -- 1.2 Computers and the Internet in Industry and Research -- 1.3 Hardware and Software -- 1.3.1 Moore's Law -- 1.3.2 Computer Organization -- 1.4 Data Hierarchy -- 1.5 Machine Languages, Assembly Languages and High-Level Languages -- 1.6 C++ -- 1.7 Programming Languages -- 1.8 Introduction to Object Technology -- 1.9 Typical C++ Development Environment -- 1.10 Test-Driving a C++ Application -- 1.11 Operating Systems -- 1.11.1 Windows-A Proprietary Operating System -- 1.11.2 Linux-An Open-Source Operating System -- 1.11.3 Apple's OS X -- Apple's iOS for iPhone®, iPad® and iPod Touch® Devices -- 1.11.4 Google's Android -- 1.12 The Internet and World Wide Web -- 1.13 Some Key Software Development Terminology -- 1.14 C++11 and the Open Source Boost Libraries -- 1.15 Keeping Up to Date with Information Technologies -- 1.16 Web Resources -- 2 Introduction to C++ Programming -- Input/Output and Operators -- 2.1 Introduction -- 2.2 First Program in C++: Printing a Line of Text -- 2.3 Modifying Our First C++ Program -- 2.4 Another C++ Program: Adding Integers -- 2.5 Memory Concepts -- 2.6 Arithmetic -- 2.7 Decision Making: Equality and Relational Operators -- 2.8 Wrap-Up -- 3 Introduction to Classes, Objects and Strings -- 3.1 Introduction -- 3.2 Defining a Class with a Member Function -- 3.3 Defining a Member Function with a Parameter -- 3.4 Data Members, set Member Functions and get Member Functions -- 3.5 Initializing Objects with Constructors -- 3.6 Placing a Class in a

Separate File for Reusability -- 3.7 Separating Interface from Implementation -- 3.8 Validating Data with set Functions -- 3.9 Wrap-Up -- 4 Control Statements: Part 1 -- Assignment, ++ and -- Operators -- 4.1 Introduction -- 4.2 Algorithms -- 4.3 Pseudocode -- 4.4 Control Structures.
4.5 if Selection Statement -- 4.6 if...else Double-Selection Statement -- 4.7 while Repetition Statement -- 4.8 Formulating Algorithms: Counter-Controlled Repetition -- 4.9 Formulating Algorithms: Sentinel-Controlled Repetition -- 4.10 Formulating Algorithms: Nested Control Statements -- 4.11 Assignment Operators -- 4.12 Increment and Decrement Operators -- 4.13 Wrap-Up -- 5 Control Statements: Part 2 -- Logical Operators -- 5.1 Introduction -- 5.2 Essentials of Counter-Controlled Repetition -- 5.3 for Repetition Statement -- 5.4 Examples Using the for Statement -- 5.5 do...while Repetition Statement -- 5.6 switch Multiple-Selection Statement -- 5.7 break and continue Statements -- 5.8 Logical Operators -- 5.9 Confusing the Equality (==) and Assignment (=) Operators -- 5.10 Structured Programming Summary -- 5.11 Wrap-Up -- 6 Functions and an Introduction to Recursion -- 6.1 Introduction -- 6.2 Program Components in C++ -- 6.3 Math Library Functions -- 6.4 Function Definitions with Multiple Parameters -- 6.5 Function Prototypes and Argument Coercion -- 6.6 C++ Standard Library Headers -- 6.7 Case Study: Random Number Generation -- 6.8 Case Study: Game of Chance -- Introducing enum -- 6.9 C++11 Random Numbers -- 6.10 Storage Classes and Storage Duration -- 6.11 Scope Rules -- 6.12 Function Call Stack and Activation Records -- 6.13 Functions with Empty Parameter Lists -- 6.14 Inline Functions -- 6.15 References and Reference Parameters -- 6.16 Default Arguments -- 6.17 Unary Scope Resolution Operator -- 6.18 Function Overloading -- 6.19 Function Templates -- 6.20 Recursion -- 6.21 Example Using Recursion: Fibonacci Series -- 6.22 Recursion vs. Iteration -- 6.23 Wrap-Up -- 7 Class Templates array and vector -- Catching Exceptions -- 7.1 Introduction -- 7.2 arrays -- 7.3 Declaring arrays -- 7.4 Examples Using arrays.
7.4.1 Declaring an array and Using a Loop to Initialize the array's Elements -- 7.4.2 Initializing an array in a Declaration with an Initializer List -- 7.4.3 Specifying an array's Size with a Constant Variable and Setting array Elements with Calculations -- 7.4.4 Summing the Elements of an array -- 7.4.5 Using Bar Charts to Display array Data Graphically -- 7.4.6 Using the Elements of an array as Counters -- 7.4.7 Using arrays to Summarize Survey Results -- 7.4.8 Static Local arrays and Automatic Local arrays -- 7.5 Range-Based for Statement -- 7.6 Case Study: Class GradeBook Using an array to Store Grades -- 7.7 Sorting and Searching arrays -- 7.8 Multidimensional arrays -- 7.9 Case Study: Class GradeBook Using a Two-Dimensional array -- 7.10 Introduction to C++ Standard Library Class Template vector -- 7.11 Wrap-Up -- 8 Pointers -- 8.1 Introduction -- 8.2 Pointer Variable Declarations and Initialization -- 8.3 Pointer Operators -- 8.4 Pass-by-Reference with Pointers -- 8.5 Built-In Arrays -- 8.6 Using const with Pointers -- 8.6.1 Nonconstant Pointer to Nonconstant Data -- 8.6.2 Nonconstant Pointer to Constant Data -- 8.6.3 Constant Pointer to Nonconstant Data -- 8.6.4 Constant Pointer to Constant Data -- 8.7 sizeof Operator -- 8.8 Pointer Expressions and Pointer Arithmetic -- 8.9 Relationship Between Pointers and Built-In Arrays -- 8.10 Pointer-Based Strings -- 8.11 Wrap-Up -- 9 Classes: A Deeper Look -- Throwing Exceptions -- 9.1 Introduction -- 9.2 Time Class Case Study -- 9.3 Class Scope and Accessing Class Members -- 9.4 Access Functions and Utility Functions -- 9.5 Time Class Case Study:

Constructors with Default Arguments -- 9.6 Destructors -- 9.7 When Constructors and Destructors Are Called -- 9.8 Time Class Case Study: A Subtle Trap-Returning a Reference or a Pointer to a private Data Member -- 9.9 Default Memberwise Assignment.
9.10 const Objects and const Member Functions -- 9.11 Composition: Objects as Members of Classes -- 9.12 friend Functions and friend Classes -- 9.13 Using the this Pointer -- 9.14 static Class Members -- 9.15 Wrap-Up -- 10 Operator Overloading -- Class string -- 10.1 Introduction -- 10.2 Using the Overloaded Operators of Standard Library Class string -- 10.3 Fundamentals of Operator Overloading -- 10.4 Overloading Binary Operators -- 10.5 Overloading the Binary Stream Insertion and Stream Extraction Operators -- 10.6 Overloading Unary Operators -- 10.7 Overloading the Unary Prefix and Postfix ++ and -- Operators -- 10.8 Case Study: A Date Class -- 10.9 Dynamic Memory Management -- 10.10 Case Study: Array Class -- 10.10.1 Using the Array Class -- 10.10.2 Array Class Definition -- 10.11 Operators as Member vs. Non-Member Functions -- 10.12 Converting Between Types -- 10.13 explicit Constructors and Conversion Operators -- 10.14 Overloading the Function Call Operator () -- 10.15 Wrap-Up -- 11 Object-Oriented Programming: Inheritance -- 11.1 Introduction -- 11.2 Base Classes and Derived Classes -- 11.3 Relationship between Base and Derived Classes -- 11.3.1 Creating and Using a CommissionEmployee Class -- 11.3.2 Creating a BasePlusCommissionEmployee Class Without Using Inheritance -- 11.3.3 Creating a CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy -- 11.3.4 CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy Using protected Data -- 11.3.5 CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy Using private Data -- 11.4 Constructors and Destructors in Derived Classes -- 11.5 public, protected and private Inheritance -- 11.6 Software Engineering with Inheritance -- 11.7 Wrap-Up -- 12 Object-Oriented Programming: Polymorphism -- 12.1 Introduction -- 12.2 Introduction to Polymorphism: Polymorphic Video Game.
12.3 Relationships Among Objects in an Inheritance Hierarchy -- 12.3.1 Invoking Base-Class Functions from Derived-Class Objects -- 12.3.2 Aiming Derived-Class Pointers at Base-Class Objects -- 12.3.3 Derived-Class Member-Function Calls via Base-Class Pointers -- 12.3.4 Virtual Functions and Virtual Destructors -- 12.4 Type Fields and switch Statements -- 12.5 Abstract Classes and Pure virtual Functions -- 12.6 Case Study: Payroll System Using Polymorphism -- 12.6.1 Creating Abstract Base Class Employee -- 12.6.2 Creating Concrete Derived Class SalariedEmployee -- 12.6.3 Creating Concrete Derived Class CommissionEmployee -- 12.6.4 Creating Indirect Concrete Derived Class BasePlusCommissionEmployee -- 12.6.5 Demonstrating Polymorphic Processing -- 12.7 (Optional) Polymorphism, Virtual Functions and Dynamic Binding "Under the Hood" -- 12.8 Case Study: Payroll System Using Polymorphism and Runtime Type Information with Downcasting, dynamic_cast, typeid and type_info -- 12.9 Wrap-Up -- 13 Stream Input/Output: A Deeper Look -- 13.1 Introduction -- 13.2 Streams -- 13.2.1 Classic Streams vs. Standard Streams -- 13.2.2 iostream Library Headers -- 13.2.3 Stream Input/Output Classes and Objects -- 13.3 Stream Output -- 13.3.1 Output of char * Variables -- 13.3.2 Character Output Using Member Function put -- 13.4 Stream Input -- 13.4.1 get and getline Member Functions -- 13.4.2 istream Member Functions peek, putback and ignore -- 13.4.3 Type-Safe I/O -- 13.5 Unformatted I/O Using read, write and gcount -- 13.6 Introduction to Stream Manipulators --

13.6.1 Integral Stream Base: dec, oct, hex and setbase -- 13.6.2 Floating-Point Precision (precision, setprecision) -- 13.6.3 Field Width (width, setw) -- 13.6.4 User-Defined Output Stream Manipulators -- 13.7 Stream Format States and Stream Manipulators -- 13.7.1 Trailing Zeros and Decimal Points (showpoint).
13.7.2 Justification (left, right and internal).

Sommario/riassunto

For Introduction to Programming (CS1) and other more intermediate courses covering programming in C++. Also appropriate as a supplement for upper-level courses where the instructor uses a book as a reference for the C++ language. This best-selling comprehensive text is aimed at readers with little or no programming experience. It teaches programming by presenting the concepts in the context of full working programs and takes an early-objects approach. The authors emphasize achieving program clarity through structured and object-oriented programming, software reuse and component-oriented software construction. The Ninth Edition encourages students to connect computers to the community, using the Internet to solve problems and make a difference in our world. All content has been carefully fine-tuned in response to a team of distinguished academic and industry reviewers. View the Deitel Buzz online to learn more about the newest publications from the Deitels. NEW! This edition is available with MyProgrammingLab, an innovative online homework and assessment tool. Through the power of practice and immediate personalized feedback, MyProgrammingLab helps students fully grasp the logic, semantics, and syntax of programming. Note: If you are purchasing the standalone text or electronic version, MyProgrammingLab does not come automatically packaged with the text. To purchase MyProgrammingLab, please visit: myprogramminglab.com or you can purchase a package of the physical text + MyProgrammingLab by searching the Pearson Higher Education web site. MyProgrammingLab is not a self-paced technology and should only be purchased when required by an instructor.