Preface

INTRODUCTION

This book has been developed from my course notes at the University of California, Los Angeles (UCLA). My eight years of teaching experience and students feedback have been the basis for the development and approach of this book. The book was written with the following objectives in mind:

1. **Overall Focus**: Since proper design and implementation represent an important part of software development, this book's primary objective is to create a balance between object-oriented programming (OOP) and C++ by bridging the gap between design and implementation. The C++ features are mapped to object-oriented concepts with less focus on language and syntax.
2. **Structure**: Since the book is structured around object-oriented concepts, each chapter addresses a specific area in OOP. For example, the beginning chapters focus on *encapsulation* and *abstraction* and the later chapters on *design hierarchies*.
3. **Object-Oriented (OO) Methodology**: This book provides formal definitions for object-oriented concepts and describes how they relate to features in C++.
4. **Complexity**: The examples are kept simple and concise and the book intentionally avoids complicated examples. I have found that extensive and complex examples only frustrate students and interfere with their comprehension of OO design concepts.

   When appropriate, a single example is consistently developed throughout a chapter. This keeps the reader's attention focused. In addition, it allows an example to evolve and address the more complex issues.
5. **Graphical Presentations**: An emphasis is placed on using graphical presentations to amplify the text. Pictorial representation has always helped my students absorb the course material more easily than just plain text.
6. **Object-Oriented Notation**: An object-oriented notation is a vehicle for conveying the design of a system in a clear and standard manner. The book primarily uses **Booch-93** notation for its examples. Booch-93 notation was selected to teach my students one of the industry's popular notations.

   This book also describes the **Unified Modeling Language (UML)**, which combines the object modeling technique (OMT) and Booch notations. The Unified Modeling Language has been developed by Grady Booch, James Rumbaugh, and Ivar Jacobson at Rational Software Corporation. The presentation of Booch-93 and UML allows the reader to select the notation that is most appropriate for his/her design.

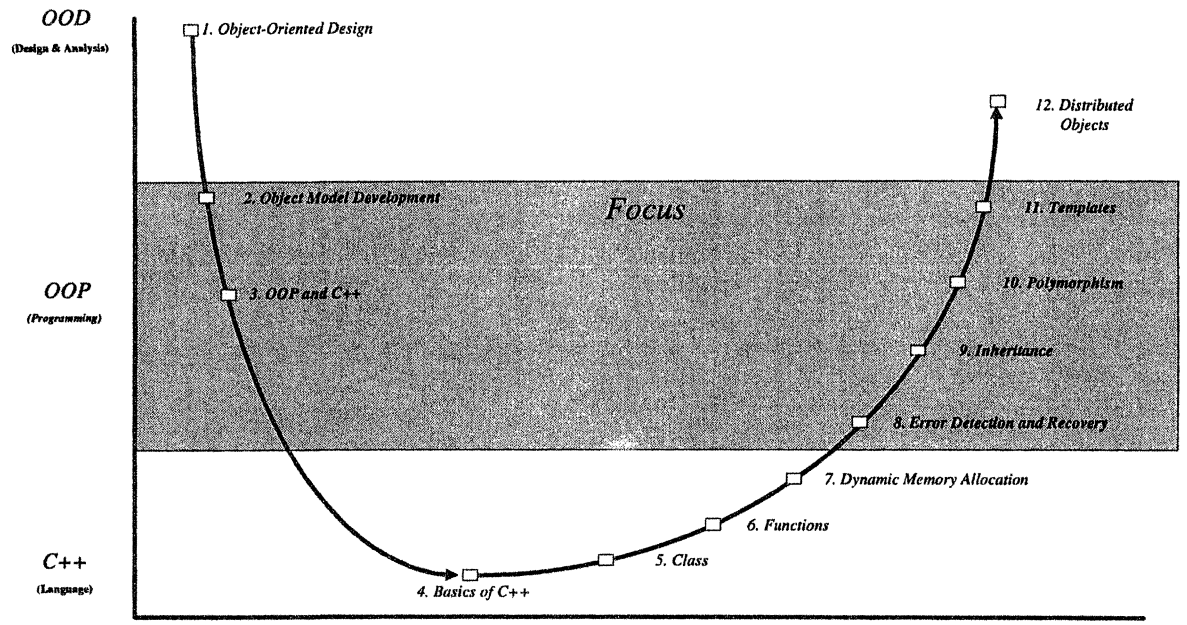
   In addition, other diagrams such as the Wassermann-Pircher-Muller and function hierarchy diagrams are also used to help convey additional information.

AUDIENCE

The text assumes the reader has prior knowledge of the C language. This assumption maintains the focus on OOP and corresponding features of C++. The book has been written for the following audiences:

1. **Students**: This textbook is designed for an upper division university course. It introduces the student to object-oriented design and programming. In addition, the student gains a basic understanding of Booch notation.
2. **Professional Software Developers and Engineers**: This book is intended for use by engineers who want to transition to OOP and C++.

FLOW



Book's flow.

The text is divided into several major segments which are identified below. Figure 1 depicts the general flow and structure of the book:

1. **Introduction**: the introduction segment begins by providing an overview of object-oriented design (OOD) and object-oriented programming (OOP) (chapters 1 through 3).

   Chapter 1 introduces the framework of object-oriented design and describes the graphical notation used.

   Chapter 2 continues to build on OOP and describes the development of an object model from requirement specification to testing. This chapter discusses the macro- and micro-development processes. It also describes how an object-oriented design captures both *dynamic* and *static* behaviors of the system.

   Chapter 3 correlates the features in C++ to the framework of an object model.
2. **C++**: To build a solid foundation on the language, Chapters 4 through 7 focus on C++. These chapters relate *encapsulation*, *abstraction*, and *modularity* from the *object model* to C++ features.
3. **Object-Oriented Programming**: Chapters 8 through 11 build on the earlier C++ work and discuss important OOP issues such as design hierarchies. These chapters describe exception handling, inheritance, polymorphism, templates, and their use in software design and implementation.
4. **Distributed Objects**: Chapter 12 introduces the reader to advanced topics such as distributed architectures. The chapter describes *concurrency* and *persistence* issues. In addition, a brief overview of the emerging standards such as the Common Object Request Broker Architecture (CORBA) standard is provided.

1. **name()**: References to function names appear in **bold face**.
2. *comments*: C++ comments in the examples appear in *italic*
3. ***data***: References to data and objects appear in ***italic-bold***.

Italic characters are also used to highlight newly introduced technical terms.
The C++ examples in this book and accompanying disk are based on the following style and layout:

```
/// FILE_NAME.H Header File
#ifndef FILE_NAME_H // multiple file inclusion guard
#define FILE_NAME_H

       #include "Supporting Library Header Filename.h"

       // #define and constant definitions
       #define SYMBOLIC_NAME constant_value
       const  Type name = initialization statement ;

       // typedef definitions
       typedef built-in_type Symbolic_Name ;

       // class definition
       class Name
       {
              private:
              // private interface: data & function declarations

              protected:
              // protected interface: data & function declarations

              public:
              // public interface: function declarations
} ;
// non-member function declarations

// inline member function definition
inline return_type Name:: function_name(parameters, if any)
{
  // function's body
}

#endif

// FILE_NAME.CPP Source File
#include <C++ Library Header Filename.h>

#include "Book's Library Header Filename.h"

// internal (static) function declarations

// initialization statements for static data members, extern, and static objects

// function definitions
return_type Class_Name:: function_name(parameters, if any)
{
   // function's body
}
```

The following identifies the coding convention in this book. An underscore is used to separate words.

1. **Data Type**: Mixed case is used for the class name and ***typedef***:

```
typedef int Counter ;

class Linked_List
{
} ;
```

2. **Constants**: Constants, #define, and enum members are defined using upper-case letters:

```
const double PI = acos( -1.0 ) ;

#define BUFFER_SIZE  100

enum Color
{
GREEN, BLUE, RED, YELLOW
} ;
```

3. **Objects**: Objects and variables are defined using lower-case letters:

```
Savings customer ;
```

ACKNOWLEDGMENTS

Writing a book is not an individual achievement. Many individuals contributed to make this book happen, and while finishing this book, I was diagnosed with a rare and advanced type of cancer in May of 1996. During these trying times, I have learned who have been my true friends. These friends have helped me spiritually, emotionally, and physically. I would like to thank especially my beautiful wife, Laura, and my friend, Patricia Dousette. In addition, Grady Booch, a true gentleman, took time away from his busy schedule to assist me in finalizing the last chapter on the Unified Modeling Language. Without their help, I would have been unable to finish this book. As one of my good friends, Alan Cordover, says: "*It does not cost a penny to be human.*" I would like to thank all of my friends for being such great humans; *thank you, my friends.*

I thank my colleagues and students, who devoted their time in proofreading the entire book, especially Patricia Dousette, Mark Turner, Wade Mergenthal, Talin Parseghian, Paul Blumstein, Mike Reese, and Paul Denzer. Their comments have been instrumental in improving the book's quality and clarity.

To lighten up a reasonably dry and technical topic, I asked Mike Reese to use his artistic ability and draw cartoons for this book. On the basis of topics presented in each chapter, Mike and I developed the cartoon themes, and Mike spent his weekends drawing them. I hope the readers will enjoy Mike's sense of humor and his cartoons.

I must thank those members of the IEEE Computer Society who played a strategic role in the publication of this book. Mohammed Fayad, Editor-in-Chief of the Practice Board, recommended this book for publication to the IEEE Computer Society. Matt Loeb, IEEE Assistant Publisher, made this book a reality, and worked closely with me in preparing it for publication. Bill Sanders, Acquisition Editor, came up with the idea for the cover of the book. Cheryl Smith and Lisa O'Conner worked diligently behind the scenes to coordinate the long and laborious process of getting the book reviewed and preparing it for publication. I thank Susan McColl, whose input improved the clarity of my book.

I am also grateful to PARTA Corp. and UCLA for helping make this book possible.

'Cause nothing' lasts forever
Even cold November rain
-Guns N' Roses
Use Your Illusion I

While this book was in its completion phase, two related significant industry developments were: the Java programming language began to gain enormous popularity, and Booch, Rumbaugh, and Jacobson consolidated the Booch, Object-Modeling Technique (OMT), Object-Oriented Software Engineering (OOSE) notations under the Unified Modeling Language (UML). Therefore, two additional chapters have been included to discuss these areas:

1. **Introduction to Java**: Chapter 13 provides a brief overview.
2. **Unified Modeling Language (UML)**: Chapter 14 concludes this book with an overview of the UML.

STANDARD C++ CLASS LIBRARIES

This book provides high-level coverage on the following ANSI C++ libraries:

1. *Stream I/O* class library: Chapters 3, 5, and 8
2. *string* class: Chapter 5
3. *complex* class: Chapter 5
4. *exception* class library: Chapter 7
5. *Standard Template Library* (STL): Chapter 10

The book describes the use of C++ features in the design of these libraries. For a comprehensive discussion, the reader should refer to a C++ compiler reference manual.

EXAMPLES

Except for the embedded application examples, the examples are available on a companion MS-DOS format disk.

FURTHER READING

This book is geared toward Booch methodology with focus on OOP and C++. The following books will further aid the reader in gaining a better understanding of object-oriented analysis/design and C++:

Booch, G. *Object-Oriented Analysis and Design with Applications.* Redwood City, California: Benjamin/ Cummings, 1994.

Stroustrup, B. *The C++ Programming Language.* New Jersey: Addison- Wesley, 1991.

Ellis, M. and Stroustrup, B. *The Annotated C++ Reference Manual.* New Jersey: Addison-Wesley, 1990.

ERRATA

I have attempted to find every error in this book. I would appreciate it if you, the reader, would notify me of any errors or omissions that you find. I also welcome suggestions for improving and enhancing the book structure and content. An errata sheet will be available with the first edition.

NOTATION

Italic and **bold face** characters are used to amplify the description of C++ examples. The following describes the convention used throughout the book: